PROCESO DESARROLLO DE HARDWARE DE LA EMPRESA BERMIT LTDA.

ANDRÉS SÁNCHEZ COMAS

Trabajo de grado presentado para optar al título de Ingeniero Electrónico

Asesor del proyecto
Esp. Rubén Darío Sánchez Dams
Ingeniero Electrónico

CORPORACIÓN UNIVERSITARIA DE LA COSTA – CUC
FACULTAD DE INGENIERÍAS
PROGRAMA DE INGENIERÍA ELECTRÓNICA
BARRANQUILLA
2011

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Barranquilla 8 de Junio de 2011

DEDICATORIA

Dedico este proyecto a Dios nuestro Señor quien nos ha dado todo los recursos, tiempo, disposición y conocimientos necesarios para la realización de este proyecto de grado y que sea el quien guíe el destino del producto de esta investigación y le de muchos aires de éxito para que pueda llegar a impactar positivamente en la sociedad y el mundo. Lo dedico también a mis padres quienes han hecho todos los esfuerzos que a su disposición han estado para apoyarme en mi carrera profesional. A Silvana por todo el amor, apoyo y motivación que me brindo durante mi carrera profesional. A la memoria de mi abuelo Carlos quien siempre celebro todos mis triunfos semestre a semestre de mi carrera y que hoy nos acompaña en nuestros corazones.

AGRADECIMIENTOS

Agradezco primero que todo a nuestro Dios, sin el esto nunca hubiera sido posible. A mis Padres por el esfuerzo y apoyo incondicional que me brindaron. A Silvana por todo el cariño, apoyo, amor, motivación y paciencia que me dio en estos cinco años de carrera profesional. A mis hermanos Kevin y Gabrielito por los momentos alegres, felices y de distracción que me brindaron en los momentos más difíciles y atareados que tuve. A mi abuela Martha a mi abuela Libia, a mi tía Viví, mi tía Nena, Rochi, mis primas Helen y Chochi quienes día a día me llenan de mucho amor, cariño y alegría aun cuando fueron muchos los días o semanas en las que no pude compartir con ellos por las ocupaciones de mi trabajo y universidad.

Agradezco enormemente a Alida, Miladys, Yisseth, Cesi, Martica, la señora Aida y Libia Subero, esas mujeres que fueron y son unas madres para mí, no olvido que durante todo el tiempo que prácticamente viví en la universidad me cuidaron, me apoyaron y se preocuparon por mí con cariño y aprecio.

Agradezco al Ing. Rubén Sánchez Dams por los años de acompañamiento académico y apoyo laboral, mi mentor de pregrado y amigo. Al Ing. Heyder Páez por la grata amistad forjada durante el pregrado y el trabajo, apoyo a momentos difíciles y por los chabacanos también. A los Ingenieros Farid Melendez y Gabriel Piñeres por el excelente acompañamiento académico y consejos de vida.

Al Sr. Mario Maury por sus excelentísimos y valiosos consejos profesionales y sobretodo enseñanzas y consejos de vida.

A Paulinia que me brindo muchos momentos de fortalezas en los momentos de cansancio y agotamiento.

En general a todos aquellos que de una u otra manera me apoyaron y cuidaron durante mi carrera profesional... a todos los quiero y les agradezco enormemente.

¡Gracias! ;)

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

Actividad: unidad tangible de trabajo realizada por un trabajador en un flujo de trabajo, de forma que implica una responsabilidad bien definida para el trabajador, produce un resultado bien definido (conjunto de artefactos), y representa una unidad de trabajo con limites bien definidos a la que probablemente se refiera el plan de proyecto al asignar tareas a los individuos.

Actores: conjunto coherente de roles que los usuarios de casos de uso desempeñan cuando interaccionan con estos casos de uso.

ALM: Aplication Life-cycle Management, herramientas para el manejo de proyectos.

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades, representa un área de responsabilidad.

Ciclo de Vida del Proyecto: fases por las que atraviesa un proyecto desde su inicio hasta su fin.

Clase: descripción de un conjunto de objetos que comparten los mimos atributos, operaciones, relaciones y semántica.

CMS: Content Managenet System, herramientas gestoras de contenido.

Concurrencia: la concurrencia son dos o más actividades durante el mismo intervalo de tiempo. La concurrencia puede conseguirse mediante el entrelazado o por la ejecución simultanea de dos o más hilos.

Dominio del problema: conjunto de conocimientos o actividad realizada sobre el que se define el problema, generalmente el que debe ser resuelto por el sistema. El dominio del problema es comprendido por lo general, por el cliente del sistema.

Dominio: área de conocimiento o agrupación de conceptos que es realizada por un conjunto de actividades con objetivos afines similares o idénticos.

Escenario: secuencia especifica de acciones que ilustran un comportamiento.

Hito: El hito principal es el punto en el que han de tomarse importantes decisiones de negocio.

Instancia: manifestación concreta de una abstracción; en Programación Orientada a Objetos es una entidad sobre la que pueden aplicarse un conjunto de operaciones y que tiene un estado que almacena los efectos de las operaciones; un sinónimo de objeto.

Mensaje: en Programación Orientada a Objetos es una comunicación entre objetos que lleva información con la expectativa de que ella

seguirá alguna actividad; la recepción de una instancia de mensajes es normalmente considerada una instancia de nuevo.

Metamodelo: Modelo o notación que define el lenguaje para expresar un modelo.

Modelo de casos de uso: Es un artefacto, el modelo de casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos.

Modelo: representación de procesos o sistemas que conforman un supra-sistema.

Riesgo: Es la variable de un proyecto que pone en peligro o impide su éxito. Los riesgos pueden consistir en que un proyecto experimente sucesos no deseados, como retrasos en la programación, desviaciones de costes o una cancelación definitiva. Los riesgos no técnicos son los relacionados con la gestión y con aspectos como los recursos (personas) disponibles, sus competencias, o con las fechas de entrega. Los riesgos técnicos son los relacionados con los artefactos de ingeniería y también con aspectos como las tecnologías de implementación, la arquitectura o el rendimiento.

Rol: Papel que cumple un trabajador. Rol que puede ser asignado a una persona o equipo, y que requiere responsabilidades y habilidades tales como realizar determinadas actividades o desarrollar determinados artefactos.

Subsistema: agrupación de elementos, de los que alguno constituyen una especificación del comportamiento ofrecido por los otros elementos contenidos.

Trabajador:

Traza: Es una dependencia que indica una relación histórica o de proceso entre dos elementos que representan el mismo concepto, sin reglas específicas para derivar una de la otra.

Usuario: El termino usuario no solo hace referencia a usuarios humanos sino a otros sistemas. En este sentido, el termino usuario representa alguien o algo (como otro sistema fuera del sistema en consideración) que interactúa con el sistema que estamos desarrollando.

Versión del producto: Conjunto de artefactos relativamente completo y consistente, del producto final que se está desarrollando y que puede ser entregado al cliente.

INTRODUCCION

Durante la academia el autor del presente proyecto de grado, presenció y vivió que los desarrollos de proyectos tecnológicos que los estudiantes de Ingeniería Electrónica desarrollan durante su formación de pregrado, están enmarcados de manera intuitiva en lo que a nivel mundial es conocido como una metodología de proyectos en cascadas, la cual comprende básicamente una secuencia lineal de actividades en bloques, que van desde el planteamiento de la idea, la planeación y el diseño para luego ser implementado, creando en la gran mayoría de los proyectos un colapso a nivel de resultados al no lograr lo que se quería, incurriendo en sobrecostos y pérdida de tiempo y esfuerzos, todo esto debido generalmente al querer diseñar y construir el sistema como un todo. El presente proyecto de grado tiene como base temática el desarrollo de proyectos tecnológicos visto desde el contexto iterativo e incremental ampliamente usado hoy en día por procesos de desarrollo de proyectos de IT como RUP, OpenUP, ICONIX y Scrum entre otros; los cuales basan su desarrollo en una serie de actividades sucesivas que se repiten durante el ciclo de vida del proyecto de veces durante el ciclo de vida del proyecto, y que buscan desarrollar el sistema como una construcción sucesiva de partes que a medida que se construyen incrementan las distintas funcionalidades que el sistema debe tener. El producto de este trabajo de grado estuvo orientado a describir y consolidar el proceso de desarrollo de sistemas embebidos de la empresa Bermit Ltda., en la base de conocimiento EPF Composer (aplicación informática especializada para la descripción de procesos de desarrollos tecnológicos), el cual fue gestado a partir del estudio de las metodologías iterativas e incrementales mencionadas anteriormente, así

como de la experiencia construida en el campo del desarrollo de proyectos tecnológicos.[1]

[1] La presente monografía fue realizada con base en la norma NTC-1486 para la Presentación de Tesis, Trabajos de Grado y otros Trabajos de Investigación, la cual faculta a partir de la tabla de contenido, la impresión a doble cara (Item 5.1.1) con márgenes simétricos a 3cm (Item 5.1.2). Cuida el medio ambiente.

1. PLANTEAMIENTO DEL PROBLEMA

Se puede observar que durante el pregrado, se enseñan principalmente conceptos y fundamentos de tecnología, mas sin embargo, estos no son suficientes para un entorno empresarial en donde se integran todos los conocimientos que durante cinco años de carrera profesional se forjan en la mente de los profesionales, esto sumado a los nuevos retos que deben enfrentar en este sector que exige resultados concretos, hace que los proyectos de desarrollo tecnológico empresariales no puedan ser llevados a cabo de la manera empírica como la que los estudiantes de Ingeniería están acostumbrados a concebir en el pregrado, ya que lo que se quiere en el ámbito empresarial es prestar un servicio de calidad en la que se busca principalmente la satisfacción del cliente y el manejo eficiente de recursos. Dicho factor de calidad en los proyectos de desarrollo tecnológicos está garantizado en gran parte por la forma en que se ejecutan los proyectos: costos lo más bajo posible, en el menor tiempo posible, y con resultados orientados al cliente.

La experiencia que la empresa Bermit Ltda. ha forjado a través de los años ejecutando proyectos y trabajando de la mano y conjuntamente con su filial en Perú en desarrollos tecnológicos a nivel regional, y sumado a la experiencia en implementación de sistemas de automatización influenciados por los Procesos de Desarrollo de Software como RUP (Rational Unified Process)[2] [1], ICONIX[3] [2], y otros como los

[2] Proceso de desarrollo de software mundialmente reconocido que se describirá más adelante en el presente trabajo de grado.

[3] Metodología de Desarrollo de Software especializado en el modelamiento y programación orientado a objetos dirigido por casos de uso de la notación UML

lineamientos de PMBOK[4] [3] (Project Management Body of Knowledge), ha llevado a esta empresa a construir un modelo de desarrollo de proyectos de hardware que ha resultado ser eficaz en este tipo de proyectos electrónicos. La debilidad está en que todo este conocimiento no está documentado sino que reside en la mente de los Ingenieros de la empresa, por lo que se hace necesario plantearlos de forma explicita y consolidarlos en una base de conocimientos que permita a la empresa gestionar mejor los proyectos, preparándose de esta manera inclusive para el ingreso de nuevo personal de Ingeniería, ya que esto lograría que la comprensión del proceso de desarrollo de proyectos de la empresa para las nuevas contrataciones no sea traumático o limitado ni dependiente del conocimiento de algunos trabajadores. Por todo lo anterior surge la pregunta, ¿De qué manera se puede formalizar la Metodología de Desarrollo de Proyectos de la empresa Bermit Ltda.?

[4] Estándar internacional para el administración de proyectos desarrollado por PMI (Project Management Institute)

2. JUSTIFICACION

En el mundo existen una gran cantidad de documentos públicos de procesos de desarrollos tecnológicos, lo que le ha permitido a la empresa BERMIT Ltda. interiorizar en la búsqueda del mejoramiento de sus procesos de desarrollo de tecnología, como por ejemplo el perfil de RUP SE (Rational Unified Process for Systems Engineering), el cual tiene tendencias al desarrollo de sistemas tecnológicos integrados por software y hardware pero mayoritariamente al desarrollo de grandes sistemas de software y no especifico al desarrollo de hardware como tal. Los estudios realizados por Bermit en esta metodología arrojan que está a pesar de su gran tamaño y complejidad puede ser adaptada a todo tipo de empresas ya sea grande, mediana o pequeña, sin embargo el costo de análisis y diseño para ajustar dicho proceso puede llegar a ser costoso y tedioso como lo comenta Marco Ebratt Gerente de la empresa, el cual también asegura que aunque dicho proceso es público y abierto a la comunidad profesional de todo el mundo, el énfasis de desarrollo de procesos y su divulgación se ha mantenido desde su publicación en el 2000 como un proceso de desarrollo de software, y pocos han sido los pasos que se han dado en este campo, como el caso que Marco menciona del lenguaje de especificación de sistemas avalado por la OMG[5], SysML desde el lanzamiento de su primera versión en Noviembre del 2005 [4], el cual es una notación capaz de combinar elementos del mundo físico con elementos del mundo lógico, pero que sin embargo SysML no impone una metodología o método de trabajo, es solo un modelo para representar abstracciones de sistemas.

[5] Object Management Group, Organización Internacional dedicada al mantenimiento y establecimiento de estándares de tecnología orientada a objetos.

Entonces, la empresa BERMIT Ltda, influenciada por los conceptos anteriores y por las metodologías de procesos de desarrollo de software como ICONIX , SCRUM[6], [5] y estándares PMBOK, con el fin de prestar servicios de calidad ha desarrollado una Metodología de Desarrollo de Hardware que busca implementar aplicaciones de sistemas embebidos dentro de un proceso eficiente, eficaz y con productos de calidad.

Se hace necesario entonces que esta metodología desarrollada por la empresa sea formalizada y registrada en una base de conocimiento con el fin de proteger su propiedad intelectual y salvaguardar su Know How, para que así se pueda socializar como un producto tecnológico desarrollado por la empresa y explotar entonces la utilización de dicho proceso como valor agregado a sus productos y servicios de manera que pueda proporcionarle a la empresa, una imagen innovadora que logre resaltarla sobre la competencia y alcanzar así un mejor posicionamiento en el mercado; además de disminuir el riesgo de fuga de conocimientos debido a que el mencionado proceso se encuentra inmerso en el conocimiento de los Ingenieros de la Empresa y algunos referentes bibliográficos.

Además, dentro de su política de responsabilidad empresarial y social, la empresa busca impulsar el desarrollo y mejoramiento profesional de los estudiantes de la carrera de Ingeniería Electrónica de la CUC mediante el convenio institucional de Prácticas Empresariales de dos formas: la colaboración al estudiante gestor del presente proyecto de grado, y la

[6] Marco de trabajo para la gestión de proyectos de IT basado en un proceso iterativo e incremental basado en la metodología de desarrollo ágil de software.

fácil apropiación de los conocimientos que puedan adquirir los futuros estudiantes en prácticas ya que se facilitaría el proceso de aprendizaje en la empresa teniendo ya manuales, guías y una base de conocimiento bien estructurada para el desarrollo de los proyecto en que la empresa pueda requerir de estos estudiantes.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Describir la Metodología de Desarrollo de Hardware de la Empresa Bermit Ltda. en una base de conocimiento para la gestión de proyectos, con el fin de facilitar el desarrollo de los productos tecnológicos de sistemas embebidos de la empresa.[7]

3.2. OBJETIVOS ESPECÍFICOS

- Definir el sistema gestor del conocimiento (Knowledge Management System, KMS por sus siglas en inglés) del proceso de desarrollo de la empresa Bermit Ltda.

- Definir los componentes básicos del desarrollo de proyectos de hardware de la empresa Bermit Ltda.

- Generar la base de conocimientos del proceso de desarrollo de hardware de la empresa Bermit Ltda.

[7] Objetivo planteado en la propuesta: Formalizar la Metodología de Desarrollo de Hardware de la Empresa Bermit Ltda. en una base de conocimiento y en una herramienta de gestión de información que permita facilitar el desarrollo de los productos y servicios de la empresa.

4. DELIMITACIONES

Actualmente la empresa Bermit Ltda. Focaliza sus actividades económicas al desarrollo de proyectos tecnológicos para la Región Caribe Colombiana aplicando un modelo de contratación bajo demanda. Para cumplir con lo anterior, la empresa cuenta con cuatro Ingenieros dedicados a la coordinación y ejecución de los distintos proyectos que la empresa logre concretar con el sector productivo haciendo contrataciones del personal técnico según la necesidad del proyecto. Así, a través de los distintos desarrollos que la empresa ha realizado, se han logrado mejorar los procesos de desarrollo de dichos proyectos buscando obtener resultados óptimos en la calidad de sus productos. La empresa tiene su sede en la ciudad de Barranquilla y el presente proyecto de grado se realiza en un momento en el que se hace necesario gestionar de una mejor manera los proyectos con el fin de abarcar proyectos de mayor envergadura y complejidad para lo cual se trabajara específicamente en el área de planificación y coordinación de proyectos caracterizando y describiendo el proceso de desarrollo en cuestión.

4.1. DEMILITACION TEMPORAL

El alcance temporal de este proyecto está comprendido desde el inicio del segundo semestre del año 2010 y el la finalización del primer semestre del 2011.

4.2. DELIMITACION ESPACIAL

El proyecto limito su realización dentro de la ciudad de Barranquilla-Atlántico en la República de Colombia, ya que es la ciudad donde se localizan las instalaciones comerciales y operativas de la empresa Bermit Ltda.

5. MARCO REFERENCIAL

5.1. MARCO CONCEPTUAL

El producto final del presente proyecto de grado tiene sus orígenes en la gestión de proyectos, ya que el resultado es un proceso de desarrollo cimentado sobre los principios de esta metodología.

5.1.1. Origen de la gestión de proyectos

En la década de los 50, los primeros en identificar esta área del conocimiento tan importante hoy en día, así como lo hicieron con muchos otros adelantos tecnológicos ha sido la industria militar con el afán de solucionar y optimizar sus actividades. Debido al desarrollo de los grandes proyectos que estos llevaban a cabo se evidencio la necesidad de coordinar los equipos y las disciplinas que trabajaban de forma simultánea en la construcción de sistemas requiriendo el trabajo concurrente y la sincronización de múltiples ingenierías, lo que llevo para los años 60 [6], a los primeros pininos en la elaboración de módulos de organización y gestión para evitar los problemas que comúnmente se aparecían en los proyectos [6]:

- Sistemas disfuncionales o deficientes
- Sobrecostos en el presupuesto
- Incumplimiento en las agendas

El militar que es considerado padre de la Gestión de Proyectos es Bernard Schiever, Arquitecto de Misiles Balísticos Polaris, ya que fue

este quien desarrolló el concepto de "concurrencia" al integrar todos los elementos del plan de desarrollo del proyecto en un solo programa y presupuesto [6], ejecutándolos en paralelo y no secuencialmente con lo que consiguió reducir los tiempos de ejecución de los proyectos Thor, Atlas y Minuteman.

Figura 5.1. Factores de éxito en el ámbito de la Gestión de Proyectos según Palacios [6]

La industria automotriz fue el segundo foco de crecimiento que le dio el impulso que necesitaba esta disciplina para surgir de la manera como la conocemos, ya que es bien reconocido el potencial dado por esta industria a los procesos de gestión industrial y de producción en los cuales cae la Gestión de Proyectos. Así surgieron de esta pujante industria de ese entonces técnicas como cronograma, histograma, el concepto de ciclo de vida del proyecto y la descomposición en tareas (WBS Work Break Down Structure) [6].

Paralelamente en los años 50, Meter Norden en los laboratorios de investigación de IBM identifico la necesidad de descomponer y planificar el trabajo de desarrollo de sistemas computaciones complejos con el fin de evitar Desbordamiento de agendas, de costes y asegurar el resultado de calidad sobre el producto, características estas que son extremadamente similares a las encontradas por la industria militar mencionada anteriormente, las cuales han llegado a construir para hoy en día patrones de comportamiento comunes a todos los proyectos con bases de conocimiento que sustentan con suficiente información las practicas válidas para cualquier proyecto, logrando desarrollar metodologías que hoy en día llevan sus nombres [6].

Así entonces el nacimiento de la gestión de proyectos arrojó conclusiones obre patrones comunes resumidos en tres puntos, demostrando que [6]:

- Es posible determinar los costes de nuevos proyectos a partir de la información de antiguos proyectos.

- Existen regularidades en todos los proyectos.

- Es absolutamente necesario descomponer los proyectos en partes de decisiones pequeñas para realizar planificaciones.

5.1.2. Ámbito General de la gestión de proyectos

Cualquier proceso ya sea productivo o administrativo puede tomar la forma de proyecto, la prestación de un servicio, la fabricación de

productos, e incluso la misma organización de la empresa. Esto debido a que según Palacio y Ruata [7]:

- Son realizadas por personas.
- Emplean recursos limitados.
- Llevan a acabo estrategias de actuación.

Con lo anterior consideremos entonces un proyecto como la ejecución de un trabajo que además de requerir recursos, personas y una ejecución controlada, se desarrolla en un marco temporal preestablecido. Así, la definición clásica de proyecto según Palacio y Ruata [7] es:

"Un conjunto único de actividades necesarias para producir un resultado definido en un rango de fechas determinadas y con una asignación específica de recursos".

Lo anterior puede ser el diseño de un computador portátil, el desarrollo de un software, la implementación de una nueva línea de producción, el diseño y ejecución de una compañía de marketing, etc.

5.1.2.1. Principios de la gestión de proyectos, nacimiento de la corriente clásica o predictiva.

Según Juan Palacio [6], "La gestión de proyectos nació para ofrecer previsibilidad en la construcción de grandes sistemas, con garantías de que el producto final se obtendrá en el tiempo y con el coste previamente estimado" Así desde sus inicios la Gestión de Proyectos

empezó a determinar del producto final, ¿Que hay que construir? detallando entonces el resultado con amplios y complejos superdocumentados de requisitos, tratando de estimar fechas y costes pre-estimados, detallando un plan de proyecto y buscando identificar las nuevas tareas y recursos a utilizar, diseñando el plan de coordinación y ejecución del proyecto, todo esto buscando supervisar y coordinar la ejecución para evitar desviaciones en el plan.

Figura 5.2. Gestión de Proyectos predictiva [7]

El principal objetivo de esta tendencia "predictiva" era que todo resultara según lo previsto buscando cumplir con las agendas, los costos y la calidad, asumiendo parámetros como [6]:

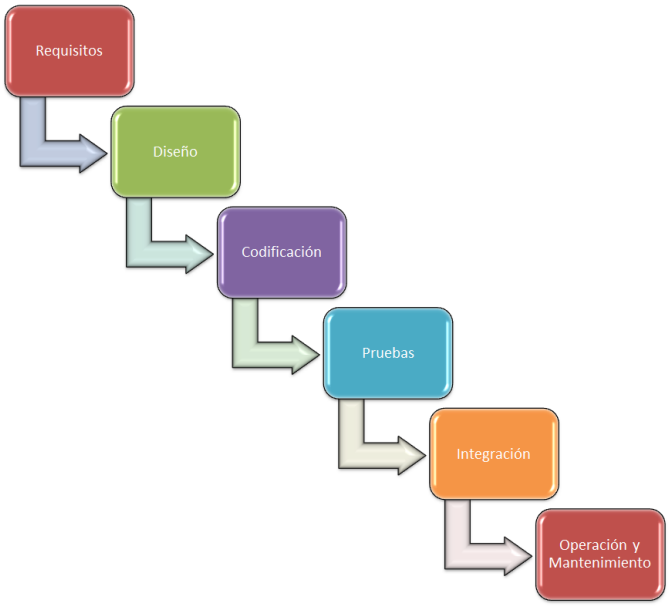
- El proyecto se desarrolla en un entorno estable y predecible.
- El objetivo es mantener el cronograma, presupuesto y recursos.

- Divide el desarrollo en fases que juntas determinan el "ciclo de vida" como concepto, requisito, diseño, planificación, desarrollo y cierre.

5.1.3. Años 80, Modelo en Cascada

Así era denominado en ese entonces esta corriente de la gestión de proyectos que buscada dividir el proyecto en fases que se ejecutaban de manera secuencial donde cada fase era realizada por personal especializado [7].

Figura 5.3. Modelo en cascada [7]



Una Fase para cada departamento con equipos especializados y profesionales especializados. Prácticamente el producto se realizaba como una carrera de relevos en la que el personal especialista pasaba el producto al personal siguiente avanzando secuencialmente a través de

las fases (creación del concepto-marketing, pruebas de viabilidad-investigación, diseño del producto-ingeniería, diseño del proceso-gestión de proyectos, desarrollo del prototipo-producción), considerándose esta corriente todavía como una tendencia predictiva [7].

5.1.3.1. Inicios de una transición

Pero para la misma época de los 80, La publicación de "The New Product Development Game"[8] [8] en el año 1986 ha marcado el punto de inicio de una nueva forma de gestionar proyectos en entornos rápidos e inestables, los cuales eran susceptibles al fracaso en la gestión de proyectos predictiva aun cuando esta estaba alcanzando una cierta madurez. Los autores del artículo observaron que algunas empresas, en mercados muy competitivos, relacionados con productos de vanguardia tecnológica, trabajaban ignorando esa teoría [7].

Esto fue en empresas norteamericanas y japonesas que producían tecnologías de primera línea con características innovadoras y menor tiempo de salida al mercado de los nuevos productos en las que Hirotaka y Ikujijo analizaron específicamente [8]:

- La fotocopiadora Fuji-Xerox FX-3500. (1978).
- La copiadora personal Canon PC-10 (1982).
- El coche urbano de 1200cc de Honda (1981).
- El ordenador personal NEC PC 8000 (1979).

[8] Título del artículo publicado en 1986 por Hirotaka Takeuchi e Ikujijo Nonaka; que a su vez daba continuación a otro de los mismos autores junto con Kenichi Imai: "Managing the New Product Development Process: How Japanese Companies Learn and Unlearn".

- La cámara Cannon AE-1 (1976).
- Cámara Cannon Auto Boy (1979).

Estas empresas no tenían un flujo de trabajo en cascada divididos por la especialidad del recurso o el personal, sino que notaron que estas contaban con un equipo multidisciplinario que trabajaba conjuntamente mediante una constante interacción desde el principio del proyecto hasta el final. Parte del resultado de esta investigación terminaría derivando en la metodología de proyectos SCRUM de la cual se habla más adelante [7].

5.1.4. Nacimiento de dos corrientes

Para el año de 1990 el modelo en cascada actuaba como una bola de nieve, a medida que transcurría el proyecto, el riesgo de fallar se hacía grande y más grande debido a la falta de retroalimentación de las partes del proceso durante el proceso, llevando a grandes impactos económicos y de esfuerzos. En contraprestación a esto nació el modelo iterativo e incremental [9] en donde el proyecto no era dividido en fases especializadas sino en partes de construcción del producto, cada parte la cual pasaba por las fases del proyecto (típicas del modelo en cascada generalmente), desarrollándose el producto del proyecto poco a poco, controlando, y previendo el desarrollo y minimizando "en tiempo real" el riesgo del proyecto.

Para la década de 1990 este modelo iterativo e incremental proveniente del Modelo en Espiral definido por primera vez por Barry Boehm en 1988 enfocado más que todo a la Ingeniería de Software [10], resultaría en

las dos grandes corrientes de metodologías de desarrollos de proyectos IT, La metodologías Agiles y la metodología tradicional. El primero representado hoy en día por una varios procesos de desarrollo entre los cuales están Scrum [5], ICONIX [2], XP [11] y Dynamic System Development Method [12] entre otros, y el segundo por su máximo exponente el Proceso Unificado.

Según los autores del Proceso Unificado de Desarrollo de Software [9] , esta metodología tradicional fomentada se identifica bajo tres características principales:

- Dirigida por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

Además de ser una metodología predispuesta a un levantamiento exhaustivo de los requisitos y control exhaustivo de la calidad enfocado a detectar defectos en las fases iniciales y reducir el número de cambios en la planeación del proyecto tanto como sea posible, intentando anticiparse a futuras necesidades buscando realizar etapas de análisis y diseño tan completas como sea posible. Mientras que las características de las Metodologías Agiles pueden verse claramente definida e identificadas en lo que es llamado el Manifiesto Ágil (ver ANEXO A).

5.1.4.1. Diferencia entre las dos corrientes metodológicas (Tradicional y Ágil) desde una perspectiva Ágil.

Figura 5.4. Comparación Desarrollo Tradicional y Desarrollo Ágil [6].

Una de las grandes características distintivas entre estas dos metodologías es que los procesos agiles no son realizados por equipos diferentes y especializados, sino por un equipo único, formado por personas muy competentes, con perfiles y conocimientos que cubren las disciplinas necesarias para llevar a cabo el trabajo [6].

En el desarrollo ágil no hay fases. En realidad las fases pasan a ser tareas que se ejecutan cuando se necesitan. No se hace primero el diseño del concepto o los requisitos, más tarde el análisis, luego el desarrollo, etc. Lo que aplicado al software serían las fases de: requisitos del sistema, requisitos del software, análisis, diseño, construcción, pruebas e integración; y se ejecutarían de forma secuencial, pasan a tareas que se llevan a cabo en el momento que hacen falta. Normalmente a lo largo de pequeñas iteraciones durante todo el desarrollo [6].

No se espera a disponer de requisitos detallados para comenzar el análisis, ni a tener éste para pasar a la codificación. Muchas veces los requisitos no se pueden conocer si no avanza el desarrollo y se va viendo y "tocando" los resultados. Otras veces el mercado es tan rápido que a mitad de trabajo las tendencias o la competencia obligarán a modificar el producto por lo que en estos casos seguir apegado a un plan de ejecución del producto seria irrisorio. Además, la participación de todo el equipo en el diseño aporta gran cantidad de talento innovador; un valor clave en el mercado de productos y servicios TIC [6].

Los equipos ágiles empiezan a trabajar sin conocer con detalle cómo será el producto final. Parten de la visión general, y sobre ella, producen regularmente incrementos de funcionalidad que incrementan el valor al producto [6].

5.1.5. Proceso Ágiles y Clásicos

A continuación detallaremos los aspectos más relevantes de los procesos que influenciaron el proceso de desarrollo de hardware de la empresa Bermit Ltda.

5.1.5.1. PU (Proceso Unificado)

Según Jacobson, Grady y Booch, autores de este Proceso de Desarrollo, este es considerado como el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software, llegando a ser un marco de trabajo genérico que puede especializarse

para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyecto. A continuación en el resto del ítem, veremos más a fondo de que se trata el Proceso Unificado según los autores del Proceso Unificado tomado del libro "El Proceso Unificado de Desarrollo de Software"[9].

Figura 5.5. Conceptualización del Proceso Unificado [9].

El proceso unificado toma como base que el sistema a construir está basado en componentes interconectados a través de interfaces bien definidas, basándose en tres principios básicos: Dirigido por Casos de Uso, Centrado en la Arquitectura, e Iterativo e Incremental.

5.1.5.2. Dirigido por los Casos de Uso

Ya que lo que se busca en el resultado final del proyecto es cumplir con los requisitos del cliente, se plasma lo que los futuros usuarios desean y necesitan en casos de uso[9] que representan a la final, los requisitos funcionales y que en conjunto constituyen el modelo de casos de uso.

[9] Es una serie sucesiva de interacciones ocurridas entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Hace parte de la notación UML de los Diagramas de Casos de Uso.

Basándose en dicho modelo y a través de un conjunto de tareas, se crean unos modelos de diseño e implementación los cuales llevan a cabo los Desarrolladores buscando ser siempre conformes con el modelo de casos de uso y la arquitectura del sistema. Así, luego de esto los ingenieros encargados prueban la implementación certificando que esté correctamente implementada la funcionalidad del caso de uso o el grupo de casos de uso que se está trabajando en el momento, es decir, la funcionalidad del cliente.

Aunque el desarrollo del proyecto es guiado por los casos de uso, esto no sucede de manera individual, estos se desarrollan a la vez que la arquitectura del sistema, siendo los casos de uso quienes guían a la arquitectura y este a su vez influye en la selección de los casos de uso.

5.1.5.3. Centrado en la Arquitectura

La Arquitectura es una perspectiva del diseño completo con las características más importantes resaltadas, dejando minuciosidades a un lado. El Arquitecto, es el rol encargado de diseñarlo, este debe centrase en objetivos adecuados como realización, capacidad de adaptación al cambio y comprensibilidad. Son los Arquitectos quienes deben trabajar sobre la comprensión general de las funciones clave del sistema, es decir sobre los Casos de Uso del sistema, los cuales suelen ser entre el 5% y 10% de todos los casos de uso, pero son los que constituyen las funcionalidades clave del sistema. Así, La forma corresponde a la Arquitectura y la funcionalidad a los Casos de Uso.

5.1.5.4. Iterativo e Incremental

Dividir el proyecto en partes más pequeñas o mini-proyectos hace el desarrollo mucho más práctico. Cada pequeño proyecto es una iteración que a la final resulta en un incremento en el proyecto. Cada iteración trata uno o más caso de uso según la necesidad y complejidad de esto, y que al ser implementados amplían la utilidad del producto que se está construyendo, así también se tratan primero los casos de uso que impliquen los riesgos más importantes, mitigando el impacto del proyecto.

A medida que se desarrollan las iteraciones los desarrolladores son quienes identifican y especifican los casos de uso relevantes creando los diseños que le dan el cuerpo al sistema seleccionado la arquitectura como guía, implementando componentes y verificando que estos satisfacen los casos de uso, es decir los requisitos funcionales del cliente, proporcionando así la arquitectura una estructura sobre la cual guiar las iteraciones, mientras los casos de uso definen los objetos y dirigen el trabajo de cada iteración.

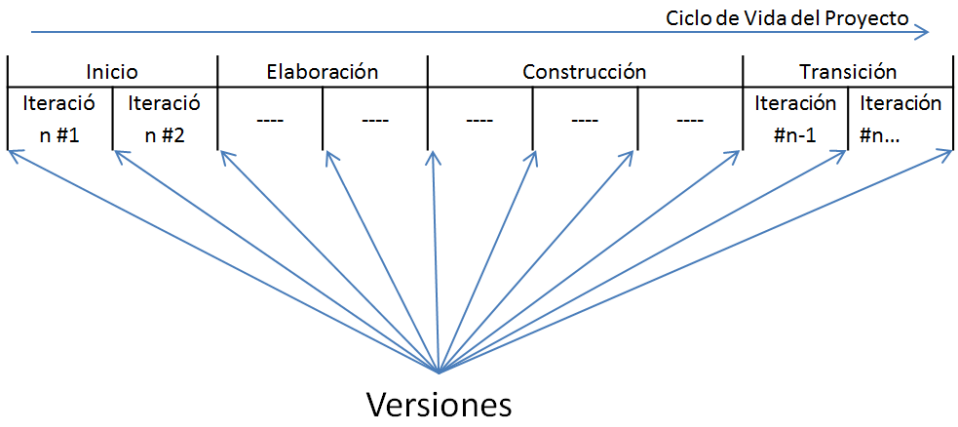
5.1.6. Partes del Proceso Unificado

Ya que el proyecto se divide en pequeñas partes o iteraciones que incrementan el desarrollo del producto, estas iteraciones se agrupan en las llamadas Fases del proceso, las cuales ejecutan iteraciones con patrones comunes en cuanto a las tareas ejecutadas, las cuales están agrupadas en Disciplinas.

5.1.6.1. Fases del proyecto

El proyecto se desarrolla a lo largo del tiempo, este tiempo se divide en cuatro fases específicas: Inicio, Elaboración, Construcción y Despliegue. Cada fase usa una secuencia de modelos los cuales visualizan el desarrollo llevado a cabo en las distintas fases, con estos los directores y desarrolladores del proyecto pueden descomponerlos en las iteraciones que implementan los casos de uso, finalizando cada fase con un hito el cual es determinado más que todo por la disponibilidad de un conjunto de artefactos cuando han alcanzado un estado predefinido.

Figura 5.6. Estructura del Proceso Unificado [9]



Durante la Fase de Inicio se desarrolla el levantamiento de la idea del producto final presentando al final de esta fase el análisis del negocio del producto en el que se tiene en cuenta las principales funciones del sistema para los usuarios finales, es decir, una idea de la arquitectura del sistema, cuanto costara el desarrollo del proyecto y cómo será el plan del ejecución.

En la Fase de Elaboración se detallan las especificaciones de la mayoría de los casos de uso del sistema y se diseña la arquitectura del sistema, al final de esta fase se espera contar con la plantificación de las actividades y la estimación de los recursos necesarios para terminar el proyecto, en la cual para la finalización de este se deberá tener en cuenta la estabilidad de los casos de uso, la definición final de arquitectura y si los riesgos están lo suficientemente controlados como para comprometer el desarrollo entero en un contrato. En la fase de construcción es donde se crea el producto, equivalente a añadir los músculos a un sistema óseo (la arquitectura).

La Fase de Transición es la que convierte al producto en una versión beta. Acá un cierto número de usuarios prueban el sistema e informan de defectos y mejoras que hay que incorporarle dirigidas a una totalidad de usuarios y que generarían una mejor versión del sistema antes de su entrega final al cliente. Es en esta fase donde se dan actividades como fabricación y capacitación del cliente para la entrega del producto.

5.1.6.2. Las disciplinas del proyecto

La categorización de las tareas dentro de RUP resulta en las llamadas disciplinas, estas recopilan las tareas relacionadas con una de las principales áreas de preocupación en el proyecto. Estas tareas están categorizadas en disciplinas que buscan cumplir objetivos comunes que se relacionen entre sí. Estas relaciones de trabajo entre las tareas se expresan en los flujos de trabajo dentro de las disciplinas. En RUP existen las llamadas Disciplinas de Ejecución y las Disciplinas de Soporte

y están dadas así:

- Modelado del Negocio.
- Requisitos.
- Análisis y Diseño.
- Implementación.
- Prueba.
- Despliegue.
- Gestión de cambios y configuración.
- Gestión de proyectos.
- Entorno.

A las disciplinas de Ejecución pertenecen Modelado del Neogocio, Requisitos, Análisis y Diseño, Implementación, Prueba y Despliegue. Estas básicamente son las que definen, diseñan y construyen el producto. Mientras que las disciplinas Gestión de cambios y Configuración y Gestión de Proyectos son las disciplinas que dan soporte a la ejecución y desarrollo del proyecto.

En la Disciplina de Modelado del Negocio a través de distintas técnica y herramientas se busca entender el entorno sobre el cual se va a trabajar con el fin de identificar mejoras potenciales, evaluar el entorno sobre el que se va a desarrollar el producto, asegurarse de que los clientes, desarrolladores, y otros interesados tengan un entendimiento común del objetivo del producto, así como también derivar o identificar los requisitos del sistemas entre otros.

En la Disciplina de Requisitos se busca obtener las solicitudes del cliente de manera explícita y transformar estos en un producto de trabajo que cubran el ámbito del sistema que se va a construir y que pueda proporcionar los requisitos detallados sobre los cuales el sistema se deberá hacer. A la final se busca establecer un acuerdo con los clientes y otros interesados de lo que el sistema debería hacer, proporcionar a los desarrolladores un conocimiento concreto de los requisitos del cliente, definir los límites del sistema (delimitarlo), proporcionar información suficiente para planificar el las iteraciones y estimar el costo del proyecto así como el tiempo de desarrollo del sistema, además de la importante tarea de definir una interfaz de usuario para el sistema que se construirá, enfocándose en las necesidades y los objetivos de los interesados y del cliente.

En la Disciplina de Diseño se procesan los productos de trabajo de los requisitos con el fin de especificar el diseño del sistema a desarrollar, acá se busca evolucionar a una arquitectura consolidada y adaptar el diseño para los ajustes a los entornos de implementación pensando siempre en obtener el máximo rendimiento del sistema.

En la Disciplina de Implementación es donde se le da forma al diseño del sistema y se busca explicar la forma en que desarrollara, se organizaran y realizaran las pruebas unitarias, y se definirá como se integraran los componentes, todo con base en los productos de la disciplina de Análisis y Diseño. En esta disciplina se organizaran, probaran y desarrollaran los componentes como unidades, y se integraran los resultados producidos por los implementadores individuales o por el equipo de implementación en un sistema ejecutable.

En la Disciplina de Pruebas se centra principalmente en la valoración o evaluación de la calidad del producto, en términos de si está o no cumpliendo con las funcionalidades de los casos de uso, es decir, de los requisitos del cliente. Acá se busca documentar los defectos del sistema, opinar sobre las mejoras en la calidad del producto, validar y demostrar las suposiciones efectuadas en las especificaciones de diseño y requisitos, con demostraciones concretas, y validar los requisitos implementados.

En la Disciplina de Configuración y Gestión de Cambios están agrupadas todas las tareas que controlan y sincronizan la evolución del conjunto del producto de trabajo que componen al sistema en construcción. Esta disciplina aporta a evitar las costosas confusiones y garantiza que los productos de trabajo resultante no tenga conflictos entre sí, ya sea por actualización simultánea, problemas de comunicación entre el equipo de desarrollo, o versiones múltiples de los componentes del sistema o del sistema como tal. También acá se gestionan los registros de contabilidad del desarrollo del proyecto y se registra la autoría de los participantes en los productos así como otros datos relevantes de este tipo concernientes al proyecto. Es por esto que esta disciplina es constantemente utilizada durante todas las fases del proyecto.
La planificación del proyecto se da en la Disciplina de Gestión de Proyectos. Es donde se proporcionan las directrices para la planificación, ejecución, supervisión, selección y gestión de personal, gestión del presupuesto, de contratos, la gestión de riesgos y la planificación de las iteraciones del proyecto.

En la Disciplina de Entorno, es donde se organizan los elementos que proporcionan el ambiente para el desarrollo del proyecto dando soporte al equipo de desarrollo abarcando desde la infraestructura hasta las herramientas.

5.1.7. ICONIX

Iconix es una metodología de desarrollo de software de tipo ágil minimalista y racionalizado cuyos esfuerzos van enfocados en ir de la mejor y más rápida forma de los casos de uso al código a través de un buen sistema de análisis y diseño. Utiliza un subconjunto esencial de la notación UML, de los 14 diagramas de que tiene UML, ICONIX utiliza solo cuatro; proporcionando requisitos suficientes y documentación de diseño sin llegar a lo que es llamado parálisis de análisis[10]. A continuación se hablara del proceso según la perspectiva de los autores en su libro "Use Case Driven Object Modeling with UML" [2].

La principal distinción de Iconix frente a los demás procesos de desarrollo es el uso del Análisis Robusto, un método para cerrar la brecha entre la etapa de análisis y el diseño. El Análisis Robusto reduce la ambigüedad en las descripciones de los casos de uso, al asegurar que estén escritos en el contexto de un modelo de dominio. Este proceso hace a los casos de uso mucho más fácil para diseñar, probar y estimar.

En esencia, el proceso Iconix describe el núcleo "lógico" del proceso de análisis y modelado de diseño. Sin embargo, el proceso puede ser

[10] Situación en la que un análisis o grupo de analistas intenta modelar y descubrir todos y a cada uno de los problemas a niveles granulares en un desarrollo informático.

utilizado sin mucha adaptación en proyectos que siguen las directrices de proyectos diferentes o metodologías ágiles. El proceso Iconix se divide en cuatro etapas:

5.1.7.1. Análisis de Requisitos

Se realiza un levantamiento de los requisitos del cliente a medida que se modela el negocio. Básicamente se construye un glosario y un modelo del dominio que definen los términos en los que el cliente y los usuarios describen el entorno sobre el cual correrá el sistema a construir. También se describen los procesos del negocio a los cuales se les adjudicaran los requisitos que se vayan definiendo durante estas actividades, buscando construir en el proceso un prototipo de interfaz de usuario para orientar y capturar con mayor precisión los requisitos del cliente definiendo los casos de uso del sistema. Al final de esta etapa del proyecto se lleva a cabo el hito[11] Revisión de Requisitos, el cual consiste básicamente en actividades de revisión para asegurarse que el texto de casos de uso que se desarrolló durante esta etapa del proyecto este escrito en los términos del dominio.

5.1.7.2. Análisis y Diseño Preliminar

Una vez que los casos de uso han sido identificados, el texto puede ser descrito de la manera cómo el usuario y el sistema van a interactuar. Un análisis robusto se realiza para encontrar posibles errores en el texto de

[11] Son momentos dentro del proyecto en que ya se ha logrado cierto estado de avance que simboliza haber conseguido cierto estado de avance en el proyecto.

casos de uso, y el modelo de dominio se actualiza en consecuencia. El texto de casos de uso es importante para determinar cómo los usuarios Interactúan con el sistema previsto. También proporcionan al analista con algo para mostrar al cliente y verificar que los resultados de los análisis de los requisitos son correctos. Luego de esta etapa se desarrolla el hito de Revisión de Diseño Preliminar que ayuda a que el diagrama robusto, el modelo del dominio, y el texto de caso de uso estén sincronizados unos con otros correctamente. La revisión de diseño preliminar es el puente entre el diseño preliminar y las etapas del diseño detallado para cada caso de uso.

5.1.7.3. Diseño Detallado

Durante esta etapa el proceso Iconix usa el modelo de dominio y el texto de casos de uso para diseñar la arquitectura del sistema. Un diagrama de clases se produce a partir del modelo de dominio y el texto de casos de uso se utiliza para hacer diagramas de secuencia. Al finalizar la etapa de Diseño Detallado se realiza el hito de Revisión Crítica de Diseño, el cual busca asegurarse de que el "como" del diseño detallado concuerde con el "que" especificados en los requerimientos, revisar la calidad del diseño, revisar la continuidad de los mensajes en los diagramas de secuencia, y revisar que las clases en el diagrama de clases tengan los métodos y atributos apropiados.

5.1.7.4. Implementación.

Se identifican, describen y organizan las pruebas unitarias a partir del cual se implementara el cuerpo del sistema permaneciendo siempre a la

altura del texto de casos de uso y los diagramas de secuencia. Por último se escribe el código utilizando la clase y los diagramas de secuencia, como una guía.

5.1.8. SCRUM

Es una metodología de trabajo para el desarrollo de proyectos de IT que se basa en el principio ágil iterativo e incremental, altamente orientado al cambio en los requisitos o prioridades del cliente, logrando gestionar los proyectos de una forma mucho más ágil. A continuación se hace una introducción completa de esta metodología [6][13][14].

Esta metodología se adapta perfectamente a equipos de desarrollo pequeños permitiendo que estos trabajen de manera auto-organizada, ya que se presume que de esta manera se consiguen más rápidos y mejores resultados. Estos pequeños equipos por lo general están integrados por diferentes disciplinas profesionales que al estar auto-organizados trabajan comprometidos y auto-motivados. Precisamente la palabra Scrum viene del vocabulario del deporte Rugby, y es la figura en que los compañeros del equipo trabajan de manera amontonada y apiñada, empujando todos en la misma dirección.

En Scrum la captura de los requisitos para el producto se realiza teniendo en cuenta la visión del cliente y del usuario directamente, y es que estos están siempre inmersos durante el desarrollo del proyecto y no al inicio como la mayoría de los procesos de desarrollo como RUP o ICONIX. El equipo captura los requisitos del cliente siendo este mismo quien dicta que requisitos se deben implementar primero, esto se

registra en unas Historias de Usuario[12] , la cuales son unas sencillas tarjetas en las que se recoge de forma esquemática y en un lenguaje claro que es lo que se quiere hacer. Esas historias de usuario se reúnen en una lista de requisitos del producto que es llamada Product Backlog[13], a cada historia de usuario el cliente le asigna una prioridad y el mismo equipo es quien le asigna cuanto tiempo se demoraran cumpliendo con el requisito del cliente.

El tiempo en el cual el equipo de desarrollo se compromete a cumplir con el requisito de el cliente se enmarcan dentro de los llamados Sprint backlog[14] y estos no deben tener una duración mayor a 4 semanas y no menor a 2 semanas. Los requisitos dentro del sprint se dividen en tareas que deben procurar no ser mayores a 16 horas, permitiendo así tener a un equipo de desarrollo flexible y auto-gestionado, ya que son ellos mismos quienes se designan las tareas.

Los roles que están inmersos en Scrum se agrupan en dos grupos: Los cerdos y los pollos. Esto viene de la parodia de la Gallina le propone al Cerdo abrir un restaurante, y el cerdo le responde preguntándole como sería el nombre del restaurante, a lo cual el pollo le responde: "Huevos con Jamón". La respuesta del cerdo es una negativa rotunda alegando

[12] Tarjetas que se usan de manera didáctica entre el equipo de trabajo y el cliente donde este plasma sus requisitos de manera esquemática y en un lenguaje claro, tratando de resumir el requisito en una frase y preferiblemente con un ejemplo genérico.
[13] Conjunto de Historias de Usuarios que son priorizados para ser trabajados dentro un Sprint. Este conjunto de Historias de Usuario se consolidan en un documento con el fin de consolidar allí el retorno de la inversión del Sprint que va a ser desarrollado, así como las estimaciones del esfuerzo requerido, lo que permitirá al Product Owner ajustar la línea temporal del Sprint.
[14] El Sprint Backlog es un documento en el que el equipo de desarrollo describe como implementara las historias de usuario en el siguiente Sprint.

que en ese caso el Pollo solo estaría implicado al poner los huevos pero él estaría comprometido al colocar el Jamón.

Siguiendo lo anterior juegan el papel de cerditos:

- El Product Owner, es el responsable de llevar y representar los requisitos del cliente y es quien aporta la visión del producto. Es este quien se encarga de escribir las historias de usuario y asignarles las respectivas prioridades ubicándolas en la lista de requisitos del producto. Este puede ser ya sea un trabajador interno de la empresa del equipo de desarrollo para un nuevo producto, o en caso de que este representando al cliente. Básicamente es el responsable por la viabilidad económica del proyecto, es decir, que este debe buscar siempre lograr altos índices del retorno de la inversión.

- El Scrum Master o facilitador, tiene como papel principal liberar de obstáculos y dificultades al resto del equipo para que este pueda cumplir a cabalidad los objetivos del sprint.

- El Equipo de Desarrollo, quien tiene la responsabilidad de construir el producto. Idealmente este equipo debería ser de entre 5 y 9 personas de distintas disciplinas (analistas, diseñadores, desarrolladores, etc.)

Los Pollos son aquellos como los usuarios finales, el cliente, los vendedores del producto, y los gestores o directivos de la empresa,

quienes son los que o bien proporcionan la visión general del producto, o dan la aprobación para con los resultados del proyecto.

Esta metodología tiene como una de sus principales características la comunicación continua entre los integrantes del equipo, ya que exige que se realicen reuniones diarias a una hora predefinida que normalmente son en la mañana, para lo cual todo el equipo debe asistir puntualmente. Esta reunión debe durar alrededor de 15 minutos y se debe realizar de pie con el fin de mantener el máximo de atención y concentración en lo que se está hablando. En esta reunión se deben formular básicamente 3 preguntas claves:

- ¿Qué me falta?
- ¿Cómo lo voy a lograr?
- ¿Qué problemas has identificado que deben ser sacados a la luz para ser resueltos lo más pronto posible?

Uno de los ámbitos de admirar de esta metodología es la comunicación en tiempo real que mantiene el equipo ya que todos saben siempre que está haciendo el equipo y como está avanzando, para esto se apoya de herramientas de la propia metodología como lo es el Burndownchart[15], el cual consiste en un tablero que contiene el estado de las tareas, si están pendiente, en proceso o si ya están listas para revisión, o si ya han sido terminadas.

[15] Grafica que muestra la cantidad de del Product Backlog que están pendiente para la fecha de finalización del sprint, lo que le evidencia al equipo el avance del proyecto.

5.1.9. Lenguajes de Modelado

Booch [15], describe la importancia de modelar realizando una comparación entre la construcción de una casa para perros, la construcción de una casa familiar, y la construcción de un edificio de negocios. Al momento de construir una casa puede usted empezar con las tablas, clavos y unas cuantas herramientas, y bien puede usted iniciar la construcción y en un par de horas sin ningún plano especializado podrá tener una casa para perros totalmente funcional.

Ahora, al momento de construir una casa para su familia, también podría iniciar con muchas más tablas de madera, clavos, otros materiales más especializados, y herramientas un tanto más especializadas. Ciertamente con esto usted se tomara mucho más que un par de horas, tal vez un par de días e incluso meses. Entonces, si usted quiere construir una casa que cumpla correctamente con las necesidades y gustos de su familia y las reglamentación de construcción deberá dibujar algunos planos que le permitan proyectar de manera razonable que deberá comprar, que cantidad y si necesitara o no personal especializado. Y aunque bien podría construirlo usted solo, debe apegarse a los planos y proyecciones de costos y tiempos, su familia podría quedar muy satisfecha.

Y si usted quiere construir un edificio de negocios, definitivamente sería tonto iniciar con una pila de tablas, clavos y unas cuantas herramientas especializadas. Ya que probablemente usted estaría jugando con el dinero de otros, le exigirán forma, estética, y funcionalidad, a menudo sus clientes cambiaran de opinión en cuanto a estos. Hará lo que usted

querrá también hacer extensas planeaciones ya que el riesgo de pérdida de dinero es altísimo. Y mientras usted trabaje con las personas indicadas, incluirá las herramientas indicadas para dirigir el proceso de transformar el concepto arquitectónico en realidad, querrá balancear el deseo de sus clientes con la construcción del edificio, disminuyendo siempre el riesgo de tirar a la basura todo lo que habrán construido.

Entonces Booch [15] describe que en el desarrollo de sistemas de software muchos proyectos pretenden construir grandes edificios pero afrontando el desarrollo como si fueran a construir una casa para perros. Entonces es en los lenguajes de modelado donde podemos construir los "modelos arquitectónicos de los edificios de negocios" o sistemas de software que se quieren construir, lo que nos permiten visualizar el producto final antes de iniciar su construcción, permitiendo proyectar la reutilización de recursos y ajustar el diseño a los requisitos del cliente y a los cambios del entorno.

Este mismo autor define con base en todo lo anterior, que un modelo es una simplificación de la realidad, los cuales se construyen para que se puedan comprender los sistemas que se van a desarrollar, dichos modelos ayudan a visualizar un sistema como debe ser o como se quiere que sean, permiten especificar la estructura de comportamiento de un sistema, dan guías y luces de cómo se deben construir, y poder documentar las decisiones que se toman sobre el desarrollo en el producto final, y que al final, modelamos sistemas para comprender la complejidad de tales sistemas.

Muchos procesos de desarrollos, independientemente de la metodología, enfoque y filosofía que usen, deben plasmar o abstraer bien sea, el comportamiento del entorno sobre el cual se desarrollara y construirá el sistema, los requisitos del cliente, la arquitectura diseñada del sistema, o la evolución y desarrollo del proyecto como tal, y esto se hace sobre modelos, los cuales buscan precisamente eso, ayudar al manejo de la complejidad de grandes sistemas o cantidades de información que por lo general las personas no podrían manejar en el contexto de sus entendimientos, roles, y tareas específicas dentro del proyecto, por lo que se busca hacer más comprensible y manejable dichos volúmenes de información para el correcto desarrollo de un proyecto.

Por lo general los modelos se componen de notaciones de diagramas que buscan representar mediante una convención de lenguaje gráfico información ya sea numérica o textual. A continuación hablamos del lenguaje más ampliamente utilizado para el desarrollo de proyectos IT, UML.

5.1.10. UML (Unificated Modeling Language).

El lenguaje de modelado unificado es un lenguaje de modelado visual diseñado para construir, visualizar, documentar y especificar los métodos o procesos de un sistema software aunque es posible extenderlo para otra multitud de aplicaciones. UML es considerado ya un estándar a nivel mundial y hay que aclarar que no es un lenguaje de programación aunque muchas herramientas permiten generar código o plantillas para la construcción del código. A continuación se hará una introducción general a esta mundialmente conocida e importante

notación de modelado tomando como referente al trabajo de los autores de este estándar, Jacobson, Grady y Booch [15].

UML está compuesto por diagramas con sus respectivas notaciones y están agrupados en tres grandes grupos Diagramas de estructuras, Diagramas de Comportamiento y Diagramas de Interacción.

Los Diagramas de Estructura se concentran en modelar y definir elementos que deben existir en un sistema, de este grupo hacen parte:

- Diagramas de clases.
- Diagramas de componentes.
- Diagramas de objetos.
- Diagramas de despliegue.
- Diagramas de estructura compuesta.
- Diagrama de paquetes.

Los Diagramas de Comportamiento se enfocan en describir lo que el sistema debería hacer, es decir, cómo funciona el sistema o lo que sucede en el sistema, y para esto hace uso de:

- Diagramas de actividades.
- Diagramas de casos de uso.
- Diagramas de estado.

Los Diagramas de Interacción pueden considerarse un subconjunto de los diagramas de comportamiento ya que se focalizan en describir el

flujo de control y de datos entre los elementos del sistema que se está modelando, a estos pertenecen:

- Diagramas de secuencia.
- Diagramas de comunicación.
- Diagramas de tiempo.
- Diagramas globales de interacción o Diagrama de vista de interacción.

Par la definición del proceso de desarrollo de la empresa Bermit Ltda. Hasta el momento se utilizan los diagramas a lo largo del proceso de desarrollo.

- Diagrama de Clases: Para la construcción del modelo del dominio que busca describir la relación entre los objetos del mundo real en términos de "tiene un" y "es un". Y para el modelo de clases sobre el cual se construirá el código del sistema para el área del software.

- Diagrama de Actividades: Para modelar el negocio sobre el cual se ejecutara el sistema a construir y gracias al cual se identifican los casos de uso del negocio que sirven para identificar las mejoras que se le harán al sistema, además de buscar conocer de manera detallada el comportamiento del negocio.

- Diagrama de Casos de Uso: En este se plasman las funcionalidades que debe tener el sistema sobre los casos de usos, y con el cual se crean las relaciones de cómo se comportan los casos de uso unos con respectos a otros. Por lo general se utilizan

las relaciones "invoca" y "precede" para ver cómo interactúan los casos de usos.

- Diagrama de Paquetes: para abstraer la complejidad del sistema en la construcción de la arquitectura del sistema. Este muestra como está dividido lógicamente el sistema en agrupaciones mostrando las respectivas dependencias entre los paquetes.

- Diagrama de Secuencia: utilizados para asignar los métodos identificados durante la construcción del Diagrama Robusto, a las clases del modelo de clases. Es decir, le asigna el comportamiento debido a las clases con el fin de darle una personalidad al sistema que se está modelando.

- Diagrama de requisitos: Donde se plasman los requisitos del sistema y se construye la relación unos con otros y a partir del cual se asignan a los casos de uso del sistema, clases, y otros. Este diagrama es originario del lenguaje de especificación de sistemas SysLM (system Lenguage Modeling por sus siglas en ingles)[16].

- Diagrama Robusto, es un diagrama especial diseñado por el proceso Iconix con el fin de identificar las clases del sistema, eliminar las ambigüedades del modelo del dominio, e identificar los métodos y operaciones de las clases.

[16] SysLM es un lenguaje de modelado avalado por la OMG (Object Management Group) y actualmente esta en la V1.2. mayores detalles:
http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf

5.2. ESTADO ACTUAL (CIENTÍFICO Y TECNOLÓGICO)

5.2.1. Herramientas soporte para el Desarrollo de Proyectos

Durante el desarrollo de este proyecto de grado, se identifico que en el desarrollo de proyectos tecnológicos, además de tener una serie de tareas sucesivas, organizadas y focalizadas, es apoyado por herramientas informaticas que son capaces de brindar todo el soporte necesario para llevar de manera más eficiente la información del proyecto permitiendo documentar, consultar, proyectar, representar y procesar la gran cantidad que en un proyecto se puede llegar a generar y que ha permitido llevar de manera eficiente y óptima la ejecución de proyectos de grandes envergaduras yendo desde el manejo de la información y la gestión del personal, hasta la proyección y gerencia del proyecto. A continuación detallamos los conceptos de los sistemas informáticos generalmente usados para el manejo de proyectos.

5.2.1.1. ALM Systems (Aplication Lifecicle Management system)

Según Perz, Durn y Bernadez en su artículo "Fundamentos para un Entrono de Application Lifecycle Dirigido por Procesos" [16] , los sistemas de administración para el ciclo de vida del proyecto se basan en herramientas que facilitan el desarrollo en numerosas áreas de trabajo como la integración de la captura de requisitos, diseño de la arquitectura, codificación, prueba y otros. Estos sistemas ayudan a incrementar la productividad al poder compartir la información con todo el equipo permitiendo que los distintos roles participantes se enfoquen en el área de proyecto que le es pertinente y hacer la trazabilidad a las

demás áreas del proyecto. Incrementan la calidad del producto permitiendo llevar la información desde los esbozos de la ideas del cliente hasta el modelo detallado del producto final. Ayuda a acelerar el desarrollo simplificando la integración de la información permitiendo hacer un trabajo más colaborativo entre los integrantes del equipo de desarrollo. Disminuye los costos de desarrollo sincronizando siempre el diseño y la aplicación final. Y aumentan la flexibilidad reduciendo el tiempo en el que se adaptan las aplicaciones que soportan las nuevas iniciativas del negocio. Así también, los anteriores autores categorizan los sistemas ALM como:

- Gestores de Tareas.
- Manejos de flujos de trabajo.
- Supervisión y reportes.
- Implementación de Software.
- Análisis de requisitos.
- Gestores de requisitos.
- Diseño.
- Modelado.
- Gestores de cambios.
- Gestores de proyectos.
- Software de Pruebas.

Entre los beneficios más notorios de esta herramienta podemos encontrar el aumento en la calidad del producto, la reducción de costos y tiempos, mejoras en la planificación, aumento de las bases de conocimiento de una empresa, automatización del desarrollo del software, documentación y generación del código así como de las

pruebas y la gestión del proyecto, facilitan el uso de las distintas metodologías propias de la ingeniería del software, gestión en todas las fases del proyecto, y permiten la reutilización del software, portabilidad y estandarización.

5.2.1.2. KMS (Knowledge Management system)

Según Becerra y Saberwai en su libro "Knowledge Management Systems and Process" [17], los Sistemas Gestores de Conocimiento son aplicaciones software utilizados comúnmente en el desarrollo de proyectos para la generación de contenido de información a través de las distintas fases en la que transcurre el proyecto. Durante el desarrollo del proyecto es casi que mandatorio generar cantidades de información que van desde el modelado de los procesos del negocio, la captura de requisitos, generación de nuevo conocimiento, análisis y procesamiento de información para su utilización en distintas áreas y fases del proyecto, toma de decisiones, revisiones, aprobaciones, informes y otros, que si se manejaran de la manera tradicional, individual y descentralizada como las organizaciones están acostumbradas a realizarlo, causarían una deficiente gestión de la información tanto en los proyectos de grandes magnitudes como los sencillos, llegando a producir atrasos, pérdidas de tiempo y sobrecostos en los procesos administrativos.

La empresa Bermit Ltda., utiliza una plataforma para la Gestión del Conocimiento especializada en el desarrollo de los proyectos. Pero debido a que esta plataforma hace parte del Know How de la empresa,

no será divulgada ni descrita en este apartado. Aun así podemos describir una categorización de los KMS según [17]:

- Knowledge Applications Systems: los sistemas para la aplicación del conocimiento son plataformas o aplicaciones dentro de una organización que ayudan a utilizar el conocimiento que normalmente está en posesión de unas pocas personas al interior de una organización o grupo, y que ha sido capturado previamente para que otras personas puedan utilizarlo de manera explícita sin necesidad de entender plenamente en que consiste dicho conocimiento con el fin de cumplir con objetivos específicos.

- Knowledge Capture Systems: Sistemas de capturas del conocimiento, que soportan el proceso de recopilación y recuperación de conocimiento tácito o explicito que reside en las personas, artefactos u organizaciones.

- Knowledge Discovery systems: Sistemas para el descubrimiento del conocimiento que soportan procesos de desarrollo de nuevo conocimiento tácito o explícito de datos provenientes de grandes volúmenes de información.

- Knowledge Sharing Systems: Sistemas de socialización del conocimientos que soportan el proceso a través del cual el conocimiento es comunicado y compartido a otros individuos de la organización.

5.2.1.3. Issue Trackinng Systems

Janak [18] en su tesis define los sistemas de gestión de incidencias también denominados Trouble Ticket System o Incident ticket System, son software especializados en la gestión de tareas, seguimiento de errores, gestión y manejo de incidencias, y en general para la gestión de proyectos que pueden ir mas allá de proyectos de tipo software permitiendo adentrarse en áreas administrativas y de gestión dentro de una organización.

Según Janak [18] En el desarrollo de proyectos de IT los Issue Tracking Systems por lo general permiten:

- Compartir información con el equipo de desarrollo
- Tener una vista general del estado del proyecto
- Establecer y actualizar la importancia de las incidencias.
- Tener un histórico de cambios.
- Recordar lo que debe ser ajustado, creado, modificado, o desarrollado.
- Saber quién reportó, cuándo y qué petición, confirmación, análisis e implementación de alguna solución fue implementada.
- Que cambios han sido realizados y cuánto tiempo tomo la resolución de alguna incidencia.

5.2.2. Casos de éxito

5.2.2.1. Proyecto Arktrans

El gobierno noruego desarrolla el proyecto Arktrans, una plataforma tecnológica para el soporte de transporte multimodal para el transporte de mercancía y pasajeros. Este proyecto se viene desarrollando desde el

año 2001 con el fin de optimizar y maximizar la capacidad de transporte de mercancía y pasajeros a través de los cuatro medios de transporte más grandes del país, aire, tierra, agua y rieles. Para este proyecto se ha usado como guía en el desarrollo del proyecto al proceso RUP, realizando actividades propias de este proceso como la identificación de los roles y skateholders en el dominio del transporte, un modelo de referencia que describe los subdominios del dominio del transporte, una vista funcional que describe la descomposición de la estructura funcional de los subdominios, una vista de comportamiento describiendo los escenarios y las relaciones entre los subdominios, y una vista de información describiendo modelos de información conceptual para el transporte de carga multimodal [19].

5.2.2.2. Designing the Large Synoptic Survey Telescope's Image Processing Pipeline

Mediante el proceso ICONIX se logró realizar el mejoramiento del sistema de procesamiento de imágenes del telescopio Large Synoptic Survey Telescope ubicado al norte de Chile el pico "El Pechón" en el monte de Cerro Pachón, a 2682 metros sobre el nivel del mar. Los directores del proyecto de mejoramiento fueron los gurús en el proceso de desarrollo Iconix, Doug Rosenberg y Math Stephens. De este proceso se destacan los resultados óptimos obtenidos teniendo en cuenta que por la característica de este proceso no se podían tratar los casos de uso "clásicos" del sistema, como se acostumbraba en el desarrollo de software, sino que se enfrentaban a un sistema en tiempo real con muchos aspectos integrados y cuya complejidad iba más allá de la implementación de las interfaces de usuario, para lo cual se enfocaron

en la identificación de los "escenarios operacionales embebidos" los cuales básicamente controlaban el hardware del sistema, siendo estos invocados por los casos de usos comunes que tenían básicamente la funcionalidad del sistema en cuanto a software.

5.2.2.3. Rediseño del sistema de información del departamento de automotores de Virginia, Estados Unidos

El Departamento de Automotores del Estado de Virginia en Estados Unidos es una agencia del gobierno encargada de administrar el parque automotor del estado así como los impuestos relacionados en beneficio de los ciudadanos del estado mismo regulando también las licencias de conducciones, titulación del parque automotor, seguridad en el transporte, y la regulación de las leyes relacionadas al transporte.

Para el año 2008, cerca de 2000 empleados de tiempo completo y medio tiempo se encontraban en las labores del departamento para proveer los servicios de administración del transporte a los clientes de los 74 Centros de servicio al cliente y 40 oficinas administrativas distribuidas por la ciudad y para esta misma fecha el departamento debía administrar más de 75 millones de vehículos y cerca de 5.3 millones de licencias. Por esto con el fin de asegurar los niveles de servicios de calidad así como el sostenimiento financiero, el Departamento decidió implementar el modelamiento de sus procesos como parte del esfuerzo para aplicar reingeniería y definir los requisitos del sistema que reemplazaría el actual en ese entonces, para lo cual se escogió el proceso ICONIX para liderar este desarrollo [20].

6. DISEÑO METODOLÓGICO

El presente proyecto de grado se realizó en el marco de una investigación descriptiva-aplicada, ya que como se ha explicado anteriormente los procesos de desarrollos tecnológicos (RUP, ICONIX, SCRUM), el conocimiento y la información de la empresa de BERMIT Ltda., estaban concentrados en sus funcionarios sin ningún tipo de documentación, lo que nos llevó hacer un levantamiento y descripción de dicha información para después consolidarla en la base de conocimiento EPF Composer, y es en este momento donde la investigación aplicada entra a jugar un papel importante ya esta plataforma gestora de información y de flujos de trabajo es usada para plasmar de manera organizada dicha información y conocimiento, permitiendo facilitar el desarrollo de proyectos en ejecuciones futuras, y redundando en que el aprendizaje de dicho proceso sea mucho más eficiente y menos costoso para la empresa, además de minimizar los riesgos de dependencia del conocimiento del personal de ingeniería.

6.1. TIPO DE ESTUDIO

El estudio utilizado en este proyecto es del tipo descriptivo ya que se buscó analizar la manera en que se manifestaba el desarrollo de proyectos tanto a nivel interno de la empresa como a nivel mundial identificando los elementos y características de estos con el fin de procesar toda esa información y plasmarla de manera organizada en una base de conocimientos que gestionara la visualización de dicha información.

6.2. POBLACIÓN DE ESTUDIO

La población de estudio para la descripción del proceso de desarrollo fueron específicamente los tres Ingenieros de proyectos los cuales vienen laborando con la empresa desde ya hace más de dos años en la realización de proyectos electrónicos y quienes tenían el conocimiento del proceso de desarrollo en cuestión.

7. TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN

7.1. TÉCNICAS DE RECOLECCIÓN DE INFORMACIÓN PRIMARIA

Se realizaron numerosas charlas con los ingenieros de la empresa con el fin de estudiar el estado actual de la forma en que se desarrollan los proyectos de la empresa, los orígenes de la gestión de proyectos manejados a su interior, conocer sus experiencias, conocimientos adquiridos, sugerencias, recomendaciones, esclarecer fuentes de información, y conocer los requisitos al momento de construir la base de conocimiento del proceso de desarrollo de la empresa. Además se realizó el modelamiento de un proyecto en un diagrama de actividades de UML con el fin de identificar patrones, actividades y tareas en el desarrollo del proyecto que pudieran alimentar la base de conocimiento del proceso de desarrollo de la empresa.

7.2. TÉCNICAS DE RECOLECCIÓN DE INFORMACIÓN SECUNDARIA

Como principal fuente de información secundaria se tiene la documentación encontrada en Internet a través del cual se obtuvo la mayor cantidad de información de los procesos de desarrollos RUP, ICONIX, OpenUP, Scrum, con los cuales se enriqueció todo el contenido que se construyó en la base de conocimiento, así como también aportó para la escogencia de la base de conocimiento y la apropiación del conocimiento acerca de cómo usarla. También se consultó bibliografía pertinente acerca de algunos temas los cuales fueron sugeridos por los

ingenieros de la empresa. En el ámbito de los libros se hizo uso extensivo de "Use Case Driving Object Modeling with UML"[2] y el "Proceso Unificado de Desarrollo de Software" [9], ambos libros máximos representantes de los procesos padres de la metodología desarrollada por la empresa, el Proceso ICONIX y Rational Unificated Process RUP.

7.3. INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN PRIMARIA

Como se mencionó en la justificación y descripción del problema del presente trabajo de grado, el conocimiento del proceso de desarrollo que la empresa había desarrollado hasta el momento se encontraba en el conocimiento (en sus mentes) de los ingenieros de la empresa, por ende el uso de la entrevista para realizar la descripción de dicho proceso de desarrollo fue vital para poder documentar y luego organizar en la base de conocimiento toda la información. En este proceso se utilizaron herramientas como hoja y papel, o software editores de texto con el fin de anotar inquietudes, ideas, respuestas, tareas, recordatorios, etc.
Instrumentos de recolección de información secundaria

Se utilizaron computadores personales para las consultas bibliográficas por internet, bases de datos de consulta especializada, bases de conocimiento de otros procesos de desarrollos como OpenUP, ICONIX, RUP y Scrum, así como también libros para las consultas bibliográficas y temas específicos de la gestión de proyectos e informes para el estudio del estado del proceso de desarrollo de la empresa implícito en la documentación de proyectos en curso y realizados.

8. RESULTADOS

Dentro de los resultados medibles y concretos a la finalización de este trabajo de grado se generó una base de conocimiento en una herramienta informática ampliamente conocida en el ámbito del desarrollo de proyectos llamada EPF Composer, que permitió plasmar la metodología de desarrollo de proyectos de la empresa de manera organizada y sistematizada, describiendo los roles que juegan los integrantes de los proyectos, las actividades, las tareas, y flujogramas que permiten describir todo el proceso empresarial concernientes al desarrollo de proyectos hardware de la empresa Bermit Ltda. Con lo anterior se espera generar un impacto positivo en la curva de aprendizaje de los miembros de la empresa en cuanto al Proceso de Desarrollo Hardware de Bermit, así como también facilitar el diseño de sus productos tecnológicos y la prestación de servicios de una manera mucho más eficiente y eficaz.

8.1. PROCESO DE DESARROLLO DE LA EMPRESA BERMIT LTDA.

Este proceso está diseñado para implementaciones de sistemas embebidos utilizando desarrollo de software orientado a objetos, teniendo como referencia que el corazón del Proceso de Desarrollo Bermit en el componente software es el Proceso ICONIX, y con el soporte de muchas de sus tareas en el proceso RUP. Esto debido a que el Proceso ICONIX es un proceso minimalista enfocado a ir rápidamente de los casos de uso al código, siendo una metodología más funcional para una Pyme en el entorno colombiano Vs el proceso RUP, el cual aunque es un proceso bastante completo es demasiado robusto,

complejo y costoso para ser aplicados a proyectos de IT en Pymes colombianas del tamaño de Bermit Ltda. Cabe anotar que ambos procesos están basados en Lenguaje de Modelado Unificado (UML por sus siglas en ingles), teniendo en cuenta que RUP dentro de su proceso utiliza todos los diagramas de UML, y por otro lado ICONIX hace el modelado del proyecto más simple reduciendo el uso de 14 diagramas que componen la notación UML a 4 diagramas. En este orden de ideas, la empresa Colombiana Bermit Ltda. ha estudiado ambos procesos y adaptado sus flujos de trabajo con unas actividades específicas identificadas por la misma compañía dentro del proceso, enfocado y dirigiendo el esfuerzo al desarrollo comercial de sistemas tecnológicos que de ahora en adelante llamaremos Proceso de Desarrollo Bermit.

En la base de conocimiento se evidencian cinco grandes tipos de contenido: Disciplinas, Flujos de Trabajo, Roles, Productos y Guías. Las Disciplinas son el conjunto de Tareas que cumplen objetivos comunes dentro de una iteración. Los flujos de trabajos son básicamente diagramas que describen la manera en que interactúan las Tareas unas con otras, es decir, dictan que actividades deben desarrollarse, antes, durante y después en un momento dado durante el desarrollo de la iteración. Los Roles es el conjunto de responsabilidades que buscan objetivos comunes y afines entre sí, y que se espera sea una persona la designada para responder por dichas responsabilidades a lo largo del ciclo de vida del proyecto. No necesariamente una persona debe cumplir un solo rol durante el proyecto, de hecho es común que una persona represente más de un rol a lo largo del desarrollo. Los productos son elementos tangibles que resultan de las tareas que son ejecutadas a lo largo del proyecto como documentos de análisis, informes, diagramas,

actas, código, simulaciones, archivos, prototipos, etc.). Y las Guías son información específica las cuales contienen conocimiento necesario para dar soporte durante el ciclo de vida del proyecto como puede ser, definiciones, formatos, ayudas, conceptos, tutoriales, normatividades, reglas, etc.

A continuación ilustraremos las partes que componen al proceso de desarrollo de la empresa describiendo solo la idea principal de cada uno de estos ya que no existe autorización corporativa para publicar el contenido completo del proceso de desarrollo en mención, por tanto no sería correcto plasmar en este documento (de posible dominio público) todo lo desarrollado en la base de conocimiento producto de este proyecto ya que estaríamos dejando expuesto el Know How de la empresa.

8.1.1. Roles

En el Proceso de Desarrollo Bermit se contemplan solo siete Roles dado el tamaño de la empresa y la filosofía de desarrollo ágil que la empresa aplica:

8.1.1.1. Rol: Gerente de Proyecto

El Gerente de Proyecto facilita los scrum diarios y se convierte en responsable de la eliminación de cualquier obstáculo que es presentada por el equipo durante las reuniones. Protege el equipo asegurándose de que no se sobre-comprometan a lo que no pueden lograr en un sprint. También lleva la planificación del proyecto, coordina las interacciones

con las partes interesadas, y mantiene al equipo del proyecto centrado en el cumplimiento de los objetivos del proyecto.

8.1.1.2. Rol: Arquitecto

El arquitecto es responsable de definir la arquitectura del sistema, que incluye la toma de las principales decisiones técnicas que limitan el diseño y la implementación general del sistema.

8.1.1.3. Rol: Analista

Es la persona dentro del equipo de desarrollo que representa al cliente y los usuarios finales involucrados, obteniendo información de los interesados para entender el problema a ser resuelto, capturando requisitos y estableciéndoles prioridades.

8.1.1.4. Rol: Desarrollador de Hardware

Este rol coordina el diseño e implementación del hardware: circuitos, interfaces, conexiones, componentes electrónicos, etc, implicados en la solución del proyecto. Esto incluye determinar la mejor estructura hardware y los componentes adecuados para la creación de prototipos.

8.1.1.5. Rol: Desarrollador de Software

La persona en este rol es responsable por desarrollar una parte del software del sistema, diseñarla ajustada a la arquitectura, implementarla, hacer pruebas unitarias, e integrar los componentes que

son parte de la solución (asegurarse que responda adecuadamente a los requisitos de concurrencia), diseño de la estructura de almacenamiento de datos persistentes. También es responsable de las actividades de pruebas y registra los resultados de las mismas.

8.1.1.6. Rol: Desarrollador de Producto

Un Diseñador de productos de presentación es el encargado de crear y recrear objetos para su producción, distribución, comercialización y uso, resolviendo problemas funcionales estéticos y comunicacionales, incorporando tecnologías y utilizando nuevos materiales y a su vez creando material gráfico del producto que se incluye como parte del embalaje del mismo. Este rol también coordina el diseño de la interfaz de usuario. Esto incluye recopilar los requisitos de utilización y los diseños de interfaz de usuario candidatos a la creación de prototipos para cumplir estos requisitos. También desarrolla los materiales de formación para preparar a los usuarios para utilizar el producto.

8.1.1.7. Rol: Interesados

Este rol representa a un grupo de interés cuyas necesidades se deben satisfacer con el proyecto. Lo puede desempeñar cualquiera que esté materialmente afectado por el resultado del proyecto. Es quien representa los intereses de los Skateholders, define las características del producto y da prioridad a los Product Backlog

8.1.2. Disciplinas

En el Proceso de Desarrollo Bermit se identificaron seis Disciplinas que guían el proceso a través del ciclo de vida del proyecto en el desarrollo del producto, de las cuales una de ellas es la disciplina de Scrum. Es una disciplina de soporte para el desarrollo del proyecto en cuanto a que no propone tareas o actividades específicas para el desarrollo del producto pero contiene la dinámica de trabajo diaria del equipo de desarrollo. Cabe anotar que esta disciplina no fue desarrollada en el presente proyecto de grado ya que las limitaciones de tiempo requeridas para finalizar la investigación no permiten desarrollar con la calidad necesaria el contenido de esta disciplina. A continuación detallaremos también las Tareas que componen las distintas disciplinas.

8.1.2.1. Disciplina: Modelado del Negocio

Esta disciplina es un conjunto de diferentes tareas que se componen de técnicas, herramientas y pasos que deben ser usadas durante el esfuerzo de ingeniería del negocio.

Tarea: Construir Modelo del Dominio

Esta tarea describe como definir un conjunto de términos manejados por los actores dentro del negocio que serán necesarios para ser usados constantemente en el proyecto, especialmente en la descripción de casos de uso.

Tarea: Agrupar subsistemas en paquetes

Identificar las agrupaciones de alto nivel que constituyen la organización. Estos pueden ser departamentos, divisiones o unidades de negocio, dependiendo de lo que la terminología de su organización utiliza. Tenga en cuenta el alcance del proyecto, no tiene sentido en la exploración de los detalles de partes de la organización que están fuera de alcance

Tarea: Revisión del Modelado del Dominio

Revisar los diagramas y descripción de los escenarios del negocio con el Revisor Interno que haya sido designado y con los Interesados

Tarea: Revisar los Procesos del Negocio

Ya que la construcción del modelo del dominio se realiza una sola vez, después de la primera captura de los requisitos, y después del primer borrador de los CUN, luego de construir el modelo del dominio es importante revisar los procesos del negocio con el fin de actualizar para que las descripciones de los CUN estén sincronizadas con el modelo del dominio.

Tarea: Capturar Procesos del Negocio

Describir y Modelar los Procesos del Negocio sobre los cuales se va a desarrollar el proyecto.

Tarea: Analizar los objetivos de la Organización

Evaluar el estado actual del cliente en cuanto a su organización y describirla en términos de los procesos actuales, herramientas, habilidades del recurso humano, y ambiente de mercadeo.

Tarea: Organizar Requisitos dentro de Diagrama de Requisitos

Definir y Jerarquizar los niveles en los que pueden estar inmersos los requisitos.

Tarea: Captura de Requisitos

En esta tarea se describe cómo obtener solicitudes de los interesados sobre qué desean que proporcione el sistema.

Tarea: Asignar Requisitos a Escenarios del Negocio

Asignación de los requisitos identificados con los escenarios del negocio

Tarea: Generar Matriz de Trazabilidad

Visualizar y formular las relaciones de los requisitos con los demás elementos a través de la herramienta Enterprise Architect

Tarea: Revisión de Requisitos

Revisión de los requisitos planteados, distribuidos, relacionados y asignados a las demás partes del proyecto.

Tarea: Estudiar Producto Competencia

Realizar una búsqueda y posterior estudio de tecnologías, plataformas o sistemas que hagan o puedan hacer parte o la totalidad del sistema que hasta el momento se ha identificado construir.

8.1.2.2. Disciplina: Requisitos

Esta disciplina explica cómo obtener las solicitudes de los interesados y transformarlas en un conjunto de productos de trabajo de los requisitos que cubran el ámbito del sistema que va a crearse y proporcionen requisitos detallados sobre lo que el sistema debe hacer.

Tarea: Actualizar el Modelo del Dominio

Sincronizar el Modelo del Dominio con las nuevas clases que se hallan descubierto, descartado o modificado durante la Revision de Casos de Uso.

Tarea: Asignar Requisitos Funcionales

Asignar requisitos funcionales a los CU y objetos del dominio

Tarea: Construir un prototipo rápido del sistema propuesto

Especie de borrador del producto final con el fin de obtener más requisitos y funcionalidades del sistema propuesto. Se refiere a las pantallas y navegación entre ellas, para hacer prototipos de caja negra sin funcionalidades implementadas, simulaciones de comportamiento,

maqueta, modelos en 3D. todo esto para sacarle más ideas de lo que el cliente quiere. Esto con el fin de imaginar el concepto de lo que el cliente quiere.

Tarea: Escribir borrador de Casos de Uso

Realizar la primera descripción del comportamiento o contenido de los Casos de Uso del Sistema dentro del Modelo y Documento de Casos de Uso

Tarea: Identificar Casos de Uso del Sistema

En esta tarea se identifican los actores y los guiones de uso que dan soporte a los requisitos que se están implementando. La identificación explícita de los actores y los guiones de uso define el ámbito del sistema.

Tarea: Obtener Información del sistema que se está heredando

Recopilar toda la información necesaria del sistema que se está heredando y sobre el cual se realizara la reingeniería para cumplir con los requisitos del cliente.

Tarea: Organizar Casos de Usos

Organizarlos lógicamente en grupos y luego en un diagrama de paquetes.

Tarea: Revisión de Casos de Uso

Extender Duplicar por lo que está en Modelado del Negocio

8.1.2.3. Disciplina: Análisis y Diseño

Esta disciplina explica cómo transformar los productos de trabajo de los requisitos en los productos de trabajo que especifiquen el diseño del software que el proyecto va a desarrollar. Es en esta disciplina donde se da forma a la Arquitectura del Sistema

Tarea: Identificar Patrones de Seguridad

Durante la arquitectura inicial de un sistema, el arquitecto de seguridad es responsable de la identificación y selección de los patrones de seguridad clave que garantiza el nivel de seguridad que necesita el sistema.

Tarea: Revisión de Diseño Preliminar

La revisión de diseño preliminar ayuda a asegurarse que los diagramas robustos, el modelo del dominio, y el texto de caso de uso, concuerden unos con otros. Esta revisión es la "puerta" entre el diseño preliminar y el diseño detallado, para cada caso de uso.

Tarea: Análisis Robusto

Para pasar de los casos de uso a los detalles del diseño (y luego al código), se necesita enlazar de los casos de uso a los objetos. La técnica

que es el Análisis Robusto la cual ayuda a cerrar la brecha entre el análisis y el diseño haciendo exactamente eso. En pocas palabras, es una forma de analizar el texto del caso de uso identificando una primera aproximación del conjunto de objetos para cada caso de uso. Estos objetos se clasifican en: objetos de interface, los objetos de entidad, y objeto de control (que a menudo son más funciones que objetos).

Tarea: Refinar la Arquitectura

Se consolida toda la información de las decisiones sobre la arquitectura.

Tarea: Análisis de la Arquitectura

Esta tarea se centra en la definición de una arquitectura candidata y en la restricción de las técnicas de arquitectura que se van a utilizar en el sistema. También describe cómo documentar los elementos del subsistema y su comportamiento, así como las dependencias del subsistema.

Tarea: Actualizar la Arquitectura

Después de analizar algún CU que impacte la arquitectura hay que actualizar la arquitectura.

En esta tarea se define cuándo y cómo se realiza la revisión de una arquitectura y cómo se revisan los resultados.

Tarea: Actualizar el Modelo Estático

Esta tarea da lugar mientras se descubren nuevos objetos y atributos durante la revisión de los CU sin embargo este proceso se finiquita dentro de la iteración. Esto equivale al modelo de clases. Actualizar los diagramas de clases a nivel de análisis.

Tarea: Revisar la Arquitectura

Esta tarea da lugar durante el Diagrama Robusto, en los casos en que el Analista, el Arquitecto o el Desarrollador de Software deban consultar el Diseño de la Arquitectura lo cual no en todas las iteraciones debería darse.

Tarea: Describir la Arquitectura en Tiempo de Ejecución

Esta tarea define una arquitectura del proceso para el sistema en términos de clases activas e instancias, y la relación de éstas con los procesos y las hilos del sistema operativo.

Tarea: Analizar la Arquitectura Hardware

El análisis de la arquitectura se centra en la definición de una arquitectura candidata y en la restricción de las técnicas de arquitectura que se van a utilizar en el sistema.

8.1.2.4. Disciplina: Diseño

En esta disciplina es la antesala al desarrollo tangible del sistema, es una disciplina de preparación para construir y probar las funcionalidades del sistema que el cliente está pidiendo.

Tarea: Actualizar Modelo Estático con métodos y atributos

Actualización y el perfeccionamiento de su modelo estático (los diagramas de clase) sobre la marcha.

Tarea: Diseñar Interfaz de Usuario

En esta tarea se explica cómo llevar a cabo el diseño de la GUI haciendo énfasis en la utilización así cómo el desarrollo de un prototipo de GUI y obtener información de retorno de utilización.

Tarea: Generar Esqueleto del Diagrama de Secuencia

Asignar el comportamiento de sus clases. Finalizar la distribución de operaciones entre las clases. Mostrar en detalle cómo las clases interactúan unos con otros durante la vida útil del caso de uso.

Tarea: Generar Pruebas Unitarias

En esta tarea se describe cómo detallar las ideas de prueba dentro de un contexto específico dirigido por los elementos del destino de la prueba, es decir, se diseña la funcionalidad específica de la prueba.

Tarea: Refinar el Modelo Estático

Detallar sobre el modelo estático los cambios, modificaciones, revisiones, y pendientes que hayan quedado en el modelo estático

Tarea: Revisión Crítica de Diseño

Asegurarse de que el diseño detallado coincida con las necesidades del usuario. Revisar la calidad del diseño. Verificar la continuidad de los mensajes. Trazar mensajes entre objetos; asignarle comportamiento a las clases del sistema

Tarea: Determinar componentes candidatos

Aquellos componentes hardware altamente candidatos a ser utilizados en la arquitectura hardware del sistema son simulados con el fin de comprobar sus funcionalidades.

Tarea: Documentar Componentes descartados

Durante la selección y evaluación de componentes a utilizar se descartan componentes hardware, y la razón por la cual fueron descartados. Deben ser descartados con el fin de engrosar el conocimiento explícito de la empresa para futuros desarrollos.

Tarea: Diseñar PCB

Diseño del Circuito Impreso o (Printed Circuit Board – PCB) del subsistema.

Tarea: Validar Diseño del PCB

Asegurar un diseño continuo del producto en una presentación estética.

Tarea: Actualizar Diseño del Producto Hardware

Luego de las sugerencias, mejoras encontradas, errores detectados, y cambios realizados o proyectados, en esta tarea se actualiza o modifica el diseño de la forma del producto.

Tarea: Diseñar Casos de Prueba de Hardware

En esta tarea se describe cómo diseñar la funcionalidad específica de la prueba. En esta tarea se describe cómo detallar las ideas de prueba dentro de un contexto específico dirigido por los elementos del destino de la prueba

Tarea: Sincronizar Hardware con Software

Sincronizar los diseños de casos de prueba que se estan realizando los flujos de software y de hardware

8.1.2.5. Disciplina: Implementación

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones de diseño.

Tarea: Construir PCB

Fabricar la plaqueta con sobre la que corre el hardware del subsistema que se está diseñando.

Tarea: Escribir Código Fuente

Escribir el código fuente de las clases a medida que se implementan las pruebas unitarias de software. En esta tarea se describe cómo producir una implementación para una parte del diseño (por ejemplo, una clase, una realización de guión de uso o una entidad de base de datos), o para solucionar uno o varios defectos. El resultado son archivos nuevos o modificados de datos y de código fuente, que se conocen generalmente como elementos de implementación.

Tarea: Generar plantilla de código de clases

Generar el código que componen las clases que se van a implementar en los casos de pruebas

Tarea: Implementar Pruebas Unitarias de Software

En esta tarea se describe cómo ejecutar y evaluar un conjunto de pruebas diseñadas para validar que el componente funciona correctamente antes de que se realicen más pruebas formales en el componente

Tarea: Revisar el Código y Actualizar el Modelo de Clases

Es probable que durante la Integración de los Subsistemas y del Sistema Completo se hayan realizado cambios en el código con el fin de realizar pequeños ajustes en la ejecución del sistema y más aún en la Prueba de Aceptación del Usuario. Es por esto que al finalizar esta última se debe actualizar el modelo con la documentación del código para referenciar dichos cambios así como el porqué de los mismos.

Tarea: Implementar Pruebas Unitarias de Hardware

Desarrollar pruebas que se puedan ejecutar conjuntamente con otras pruebas como parte de una infraestructura de prueba mayor

8.1.2.6. Disciplina: Pruebas

Esta disciplina proporciona orientación sobre cómo evaluar y valorar la calidad del producto.

Tarea: Documentar Inconvenientes de Hardware

Durante la implementación de las pruebas unitarias de hardware, construcción de los PCB y montaje en protoboard, pueden presentarse dificultades los cuales puede que sean o no solucionados, generan conocimiento acerca de inconvenientes específicos que posiblemente pueden presentarse en un futuro ya sea en el mismo proyecto o en otro, de aquí la importancia de engrosar el conocimiento explícito de la empresa en el desarrollo de proyectos mediante la documentación de los inconvenientes de hardware que se puedan presentar.

Tarea: Integrar el Sistema

Integración de los subsistemas que se han implementado en el Caso de Uso que se está implementando

Tarea: Integrar los subsistemas

En esta tarea se describe cómo integrar los elementos en un subsistema de implementación y, a continuación, entregar el subsistema de implementación para su integración en el sistema.

Tarea: Planificar la integración del Sistema

Luego de superar las pruebas unitarias de hardware y software de los subsistemas que se están implementando para los casos de uso, se deben integrar estos con los demás subsistemas ya implementados. En esta tarea se planifica como será dicha integración teniendo en cuenta los casos de prueba.

Tarea: Ejecutar prueba del Sistema y Aceptación del Usuario

En esta tarea se describe cómo garantizar que el producto desarrollado cumpla sus criterios de aceptación tanto en el sitio de desarrollo como en el sitio de instalación de destino especialmente de los clientes y usuarios finales.

Tarea: Documentar Inconveniente Software

Durante la implementación de las pruebas unitarias de software y la redacción del código de las clases y código fuente, pueden presentarse

dificultades los cuales puede que sean o no solucionados, generando conocimiento acerca de inconvenientes específicos que posiblemente pueden presentarse en un futuro ya sea en el mismo proyecto o en otro, de aquí la importancia de engrosar el conocimiento explícito de la empresa en el desarrollo de proyectos mediante la documentación de los inconvenientes de software que se puedan presentar.

8.1.3. Productos

Como ya se dijo anteriormente, Los productos son elementos tangibles resultado de las tareas que son ejecutadas a lo largo del proyecto como documentos de análisis, informes, diagramas, actas, código, simulaciones, archivos, prototipos, etc. Los productos están categorizados por el dominio al cual hacen parte las Tareas que los generan, es decir, las disciplinas. A continuación listaremos los productos que han sido identificados se generan dentro del Proceso de Desarrollo Bermit y que están plasmados dentro de la Base de Conocimiento producto del presente proyecto de grado.

Modelado del Dominio:

- Glosario.
- Modelo del Dominio.
- Evaluación de la Organización Cliente.
- Plantilla RCUN.
- Consolidado Análisis Sistemas Heredados.
- Registro de Revisión.

Requisitos:

- Modelo de Requisitos.
- Modelo de Casos de Uso.
- Descripción de Casos de Uso.
- Prototipo de Interfaz de Usuario.
- Guion Grafico.
- Prototipo de Caja Blanca.
- Registro de Revisión.

Análisis y Diseño Preliminar:

- Documento Arquitectura de Prueba de Concepto.
- Documento de Arquitectura Técnica Software.
- Documento de Arquitectura Técnica Hardware.
- Registro de Revisión.

Diseño:

- Circuito PCB.
- Documento de Componentes.
- Modelo de Clases.
- Registro de Revisión.

Implementación:

- Diseño de esquemático
- Diseño del PCB
- Sistema Producto Final

- Documento de Aceptación de Iteración
- Circuito PCB
- Código de Clases
- Registro de Revisión

Pruebas:

- Caso de Prueba.
- Lista de ideas de Pruebas.
- Lista de cotización de componentes hardware.
- Diseño Protoboard.
- Documento de inconveniente de Pruebas Software.
- Plan de integración de Subsistemas.
- Registro de revisión.

8.1.4. Guías:

Dentro de la Base de Conocimiento del Proceso de Desarrollo Bermit, la información específica que contienen conocimiento necesario para dar soporte durante el ciclo de vida del proyecto se categorizan como Guías, a continuación listamos las guías que fueron identificadas para esta versión del Proceso.

- Arquitectura de componentes.
- Arquitectura del sistema.
- Arquitectura de Software.
- Bussines Architecture.
- Capas.

- Contrato Ágil SCRUM.
- Diagramas de espina de pescado.
- Diagrama de graficos de estados.
- Diagrama de Pareto.
- Entrevistas.
- Estándares de Protección de Hardware.
- Guion Grafico.
- Juegos de Rol.
- Prototipo.
- Requisitos Funcionales.
- Revisión de los requisitos existentes.
- Scope of Bussines Modeling.
- Talles de Requisitos.
- Taller de Valoración.
- Tormenta de ideas y Redacción de ideas.

8.1.5. Flujos de Trabajos

A continuación mostramos los diagramas que describen la manera en que interactúan las Tareas unas con otras. Estos son diagramas basados en el Metamodelo para la Especificación de Procesos de Ingeniería de Software y Sistemas SPEM[17] (Software & Systems Process Engineering Metamodel Specification por sus siglas en ingles), que tiene integrado la herramienta EPF Composser con la cual se construyó y definió el Proceso de Desarrollo Bermit.

[17] Mayores detalles visitar: http://www.omg.org/spec/SPEM/2.0/

Ya que estos flujos de trabajo no son más que un diagrama, la empresa considera que colocar de manera explícita dichos diagramas seria arriesgar el valor intelectual que esto representa en los activos intelectuales de la organización dado el esfuerzo que se ha hecho en desarrollar el Proceso de Desarrollo. Por ende colocaremos una captura de las imágenes que evidencien la existencia de los flujos de trabajo sin vulnerar la propiedad intelectual de la empresa. Sin embargo cabe anotar que en la sustentación del presente trabajo de grado se mostraran a los jurados, evaluadores y presentes dichos flujos de trabajos.

Figura 8.1. Modelado del negocio.

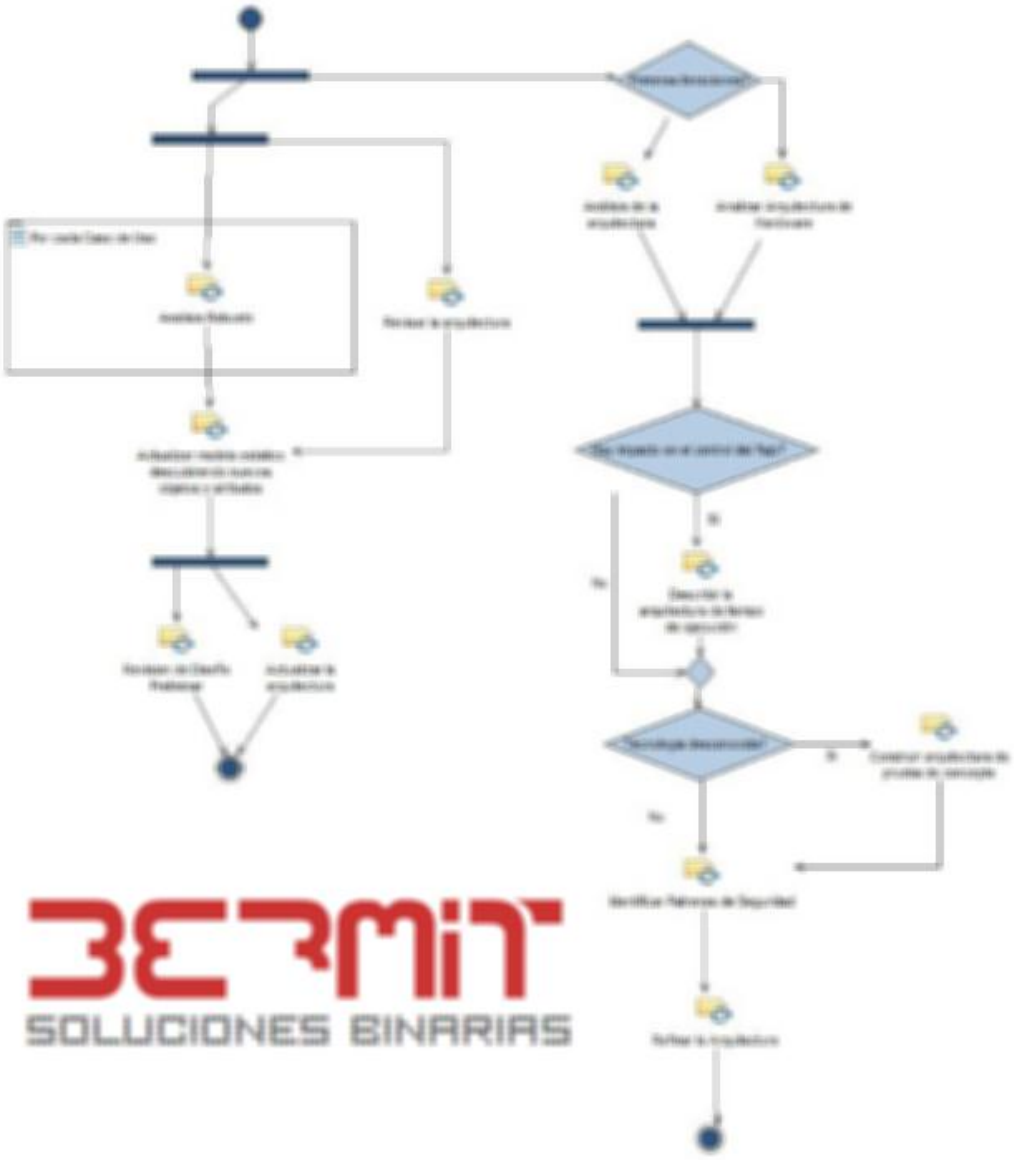


Figura 8.2. Requisitos.

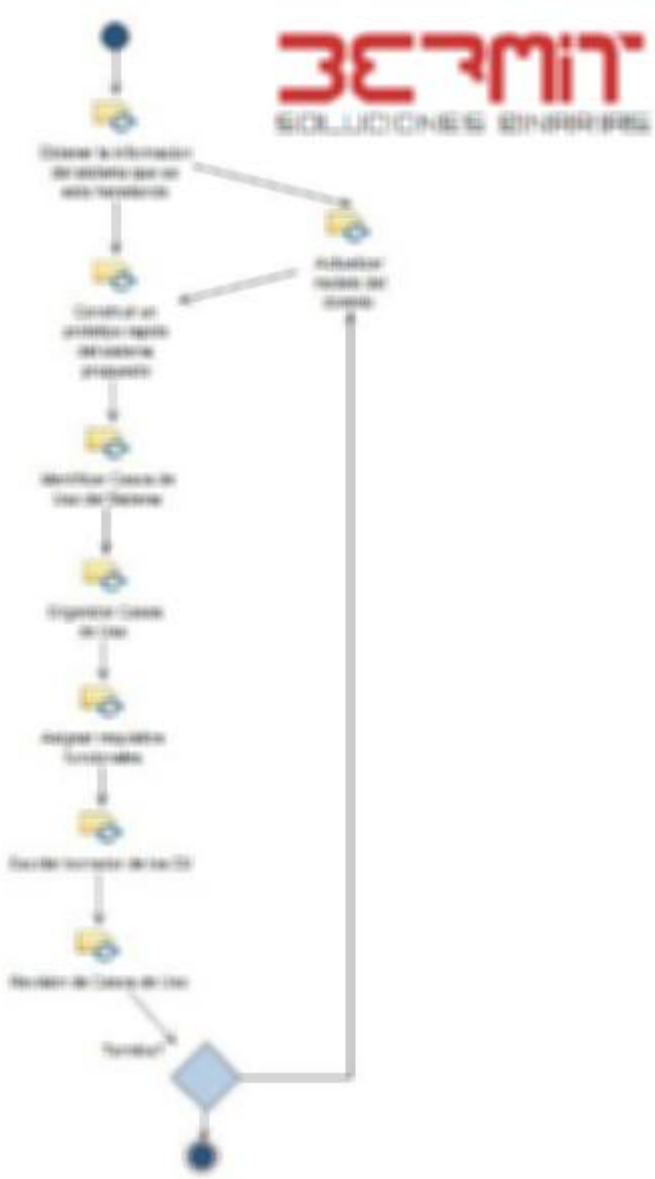


Figura 8.3. Análisis y diseño preliminar.

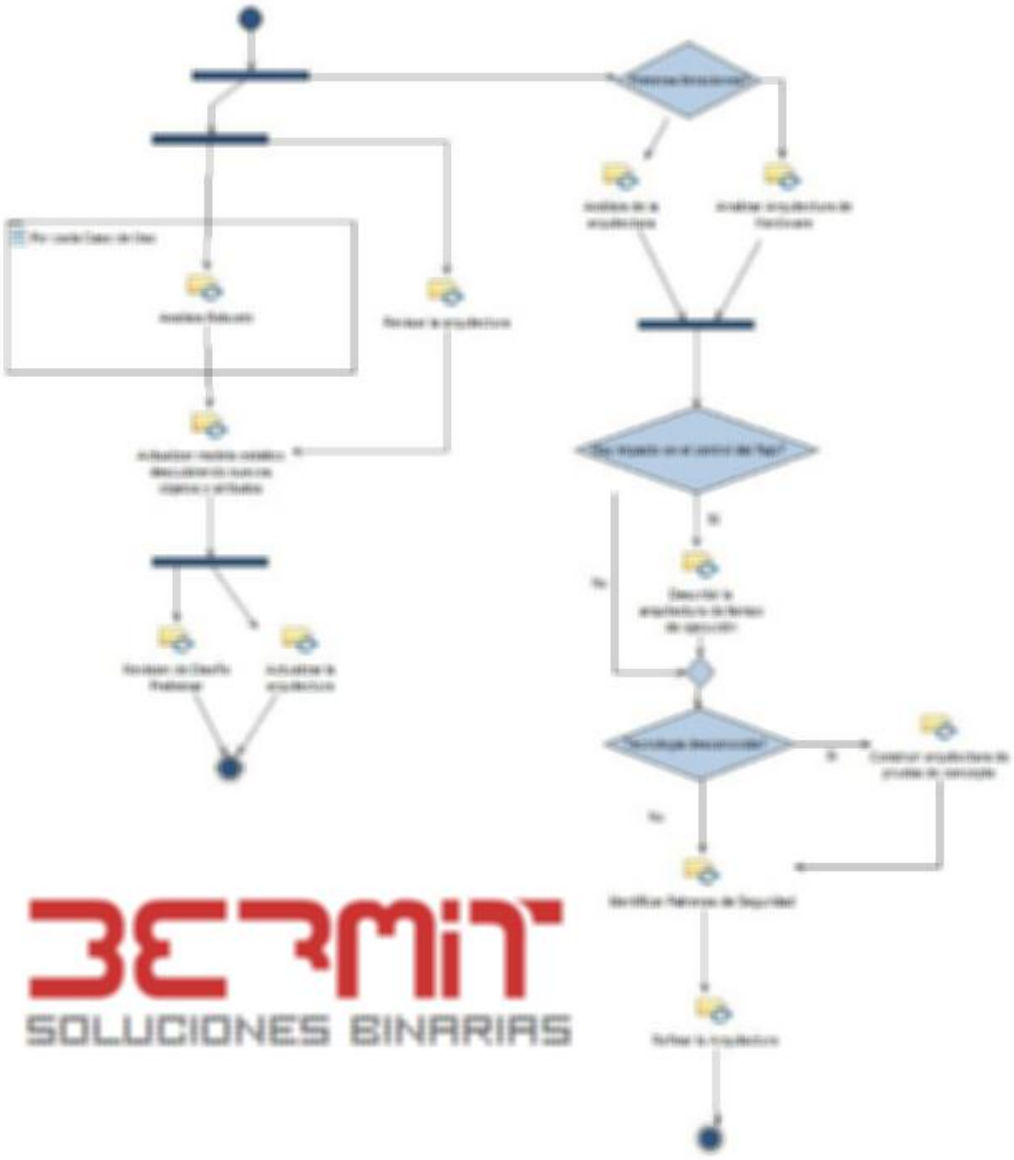


Figura 8.4. Diseño.

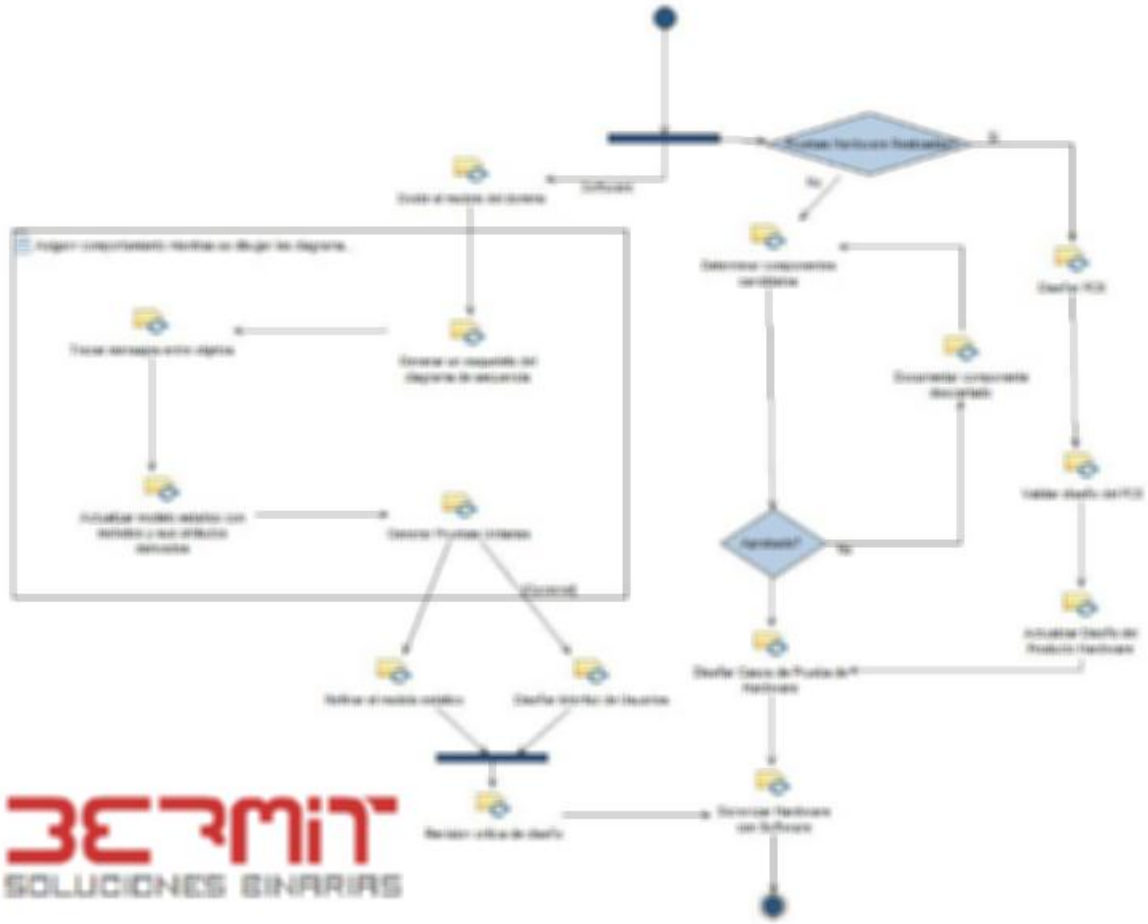
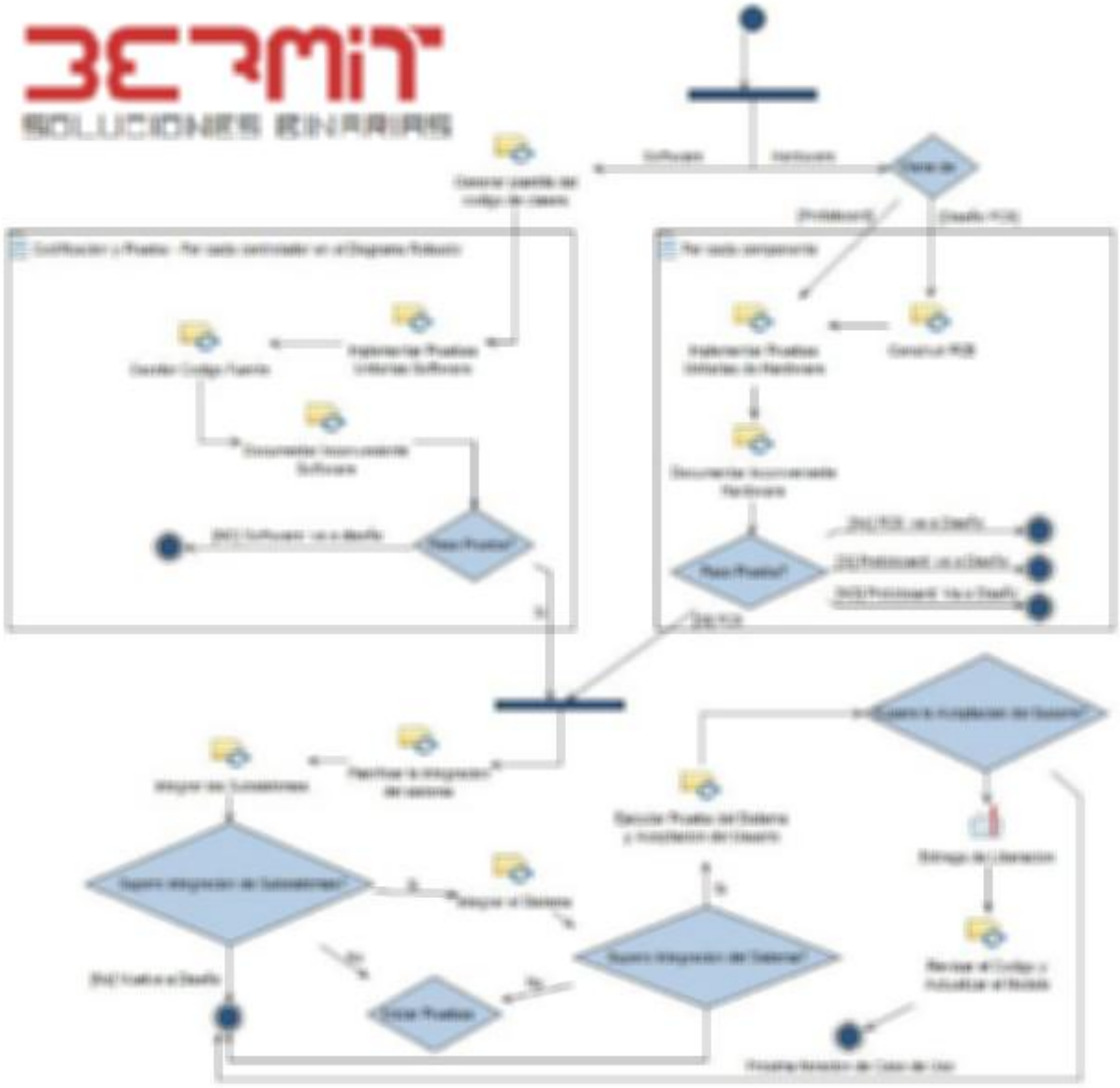


Figura 8.5. Implementación.



9. Divulgación

El resultado de este proyecto se materializa en la producción de un contenido web totalmente autogestionado y de fácil implementación en la intranet de la empresa, de manera que la información de la base de conocimiento pueda estar siempre disponible para todo el personal pertinente a esta área de conocimiento de manera rápida y eficiente gracias a la interfaz de usuario que esta herramienta genera. A continuación mostramos una captura de pantalla del contenido web ejecutado sobre un navegador web localmente.

Figura 9.1. Vista principal de la Base de Conocimiento del Proceso de Desarrollo Bermit

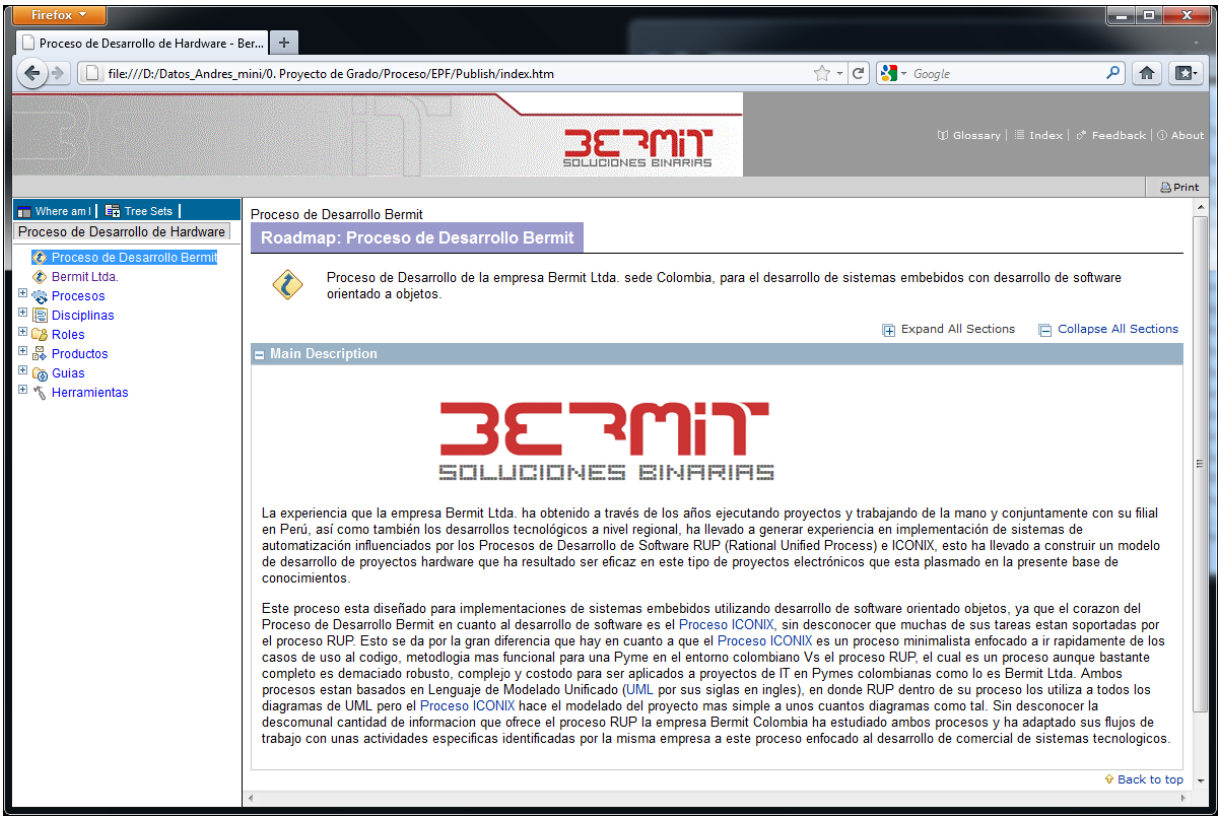
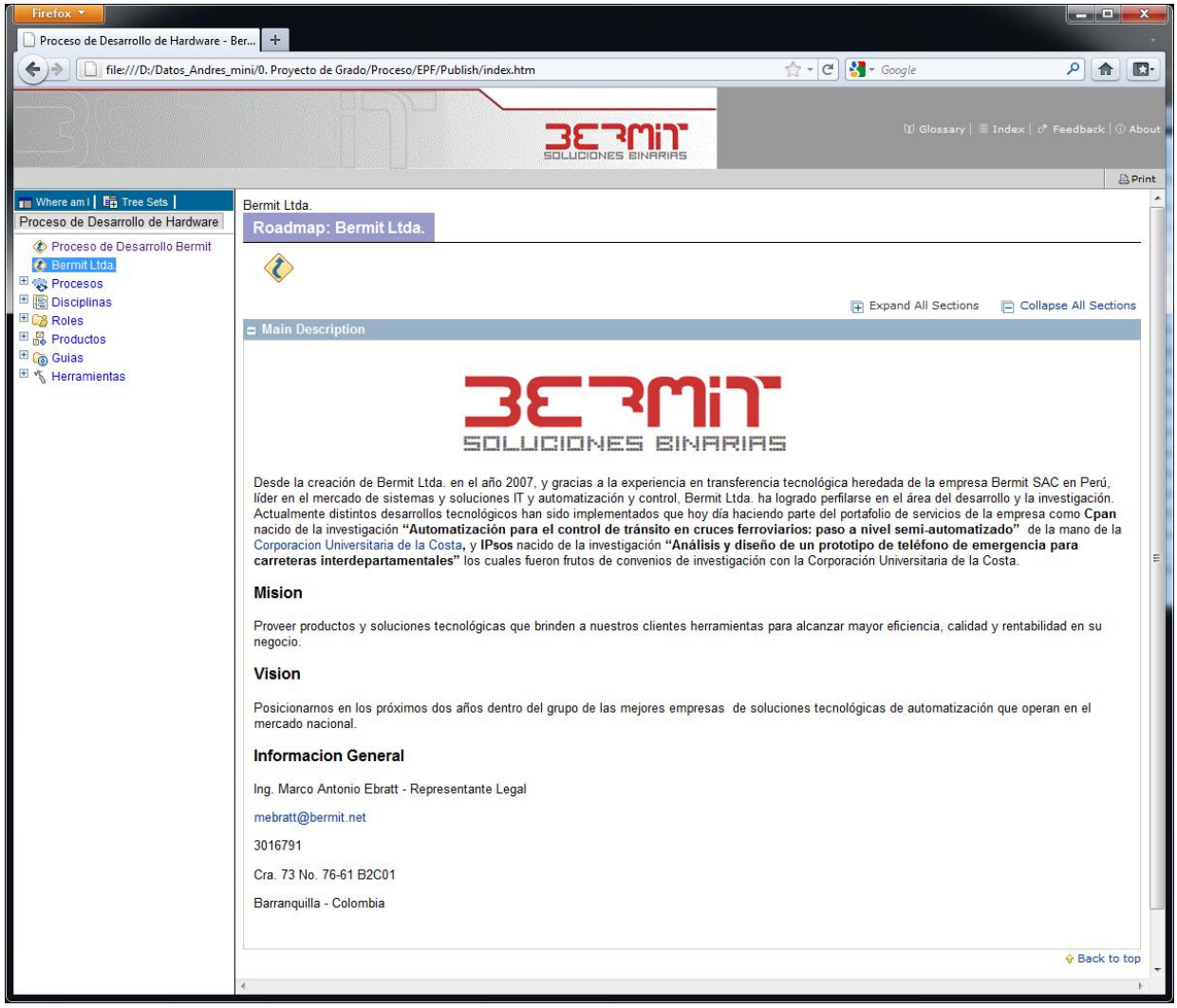


Figura 9.2. Vista alternativa de la Base de Conocimiento del Proceso de Desarrollo Bermit.



BIBLIOGRAFÍA

[1] P. Kroll y P. Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, 1o ed. Addison-Wesley Professional, 2003.

[2] D. Rosenberg y M. Stephens, *Use Case Driven Object Modeling with UML: Theory and Practice*. Apress, 2007.

[3] C. S. Stackpole, *A User's Manual to the PMBOK Guide*, 1o ed. Wiley, 2010.

[4] «SysML.org - Open Source Specification Project». [Online]. Available: http://www.sysml.org/. [Accessed: 26-Jun-2011].

[5] K. H. Pries y J. M. Quigley, *Scrum Project Management*. CRC Press, 2010.

[6] J. Palacio, *Flexibilidad con Scrum. Principios de diseño e implantación de campus de Scrum*. 2007.

[7] J. Palacio y C. Ruata, *Scrum Manager. Gestión de proyectos*. 2010.

[8] I. Nonaka y H. Takeuchi, «The New Product Development Game», *Hardvard Bussines Review Hardvard College*, págs. 137-146, Feb-1986.

[9] G. Booch, J. Rumbaugh, y I. Jacobson, *El proceso unificado desarrollo software*, 8o ed. Pearson Addison-Wesley, 2000.

[10] B. Boehm, «A spiral model of software development and enhancement», págs. 61-72, May. 1988.

[11] K. Beck, *Extreme Programming Explained: Embrace Change*, US ed. Addison-Wesley Professional, 1999.

[12] B. J. J. Voigt, *Dynamic System Development Method*. Suiza: Universidad de Zurich, 2004, pág. 16.

[13] K. Schwaber y M. Beedle, *Agile Software Development with Scrum*, 1o ed. Prentice Hall, 2001.

[14] H. Kniberg, *Scrum and XP from the Trenches*. Lulu.com, 2007.

[15] G. Booch, J. Rumbaugh, y I. Jacobson, *The Unified Modeling Language User Guide*, 1o ed. Addison-Wesley Professional, 1998.

[16] J. D. Prez–Jimnez, A. Durn, y B. Bernrdez, «Fundamentos para un Entorno de Application LifecycleManagement Dirigido por Procesos», vol. 3, n°. 3, págs. 41-48, 2009.

[17] I. Becerra-Fernandez y R. Sabherwal, *Knowledge Management: Systems and Processes*. M.E. Sharpe, 2010.

[18] J. Janák, «Issue Tracking Systems», Universidad Masarykk, 2009.

[19] SINTEF, *ARKTRANS Forprosjekt Felles, multimodal systemarkitektur for telematikk i person- og godstransport*. pág. 2001 Septiembre 15.

[20] Sparx Systems, «VIRGINIA DEPARTMENT OF MOTOR VEHICLES. Systems Redesign with Enterprise Architect.», pág. 6, May. 2011.

ANEXO A. MANIFIESTO ÁGIL

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

- El software funcionando es la medida principal de progreso.

- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

ANEXO B. CRONOGRAMA DE ACTIVIDADES

Tabla 9.1. Cronograma ejecutado durante el desarrollo de la investigación

CRONOGRAMA DE DESARROLLO DEL PROYECTO DE GRADO	2010			2011					
	Oct	Nov	Dic	Ene	Feb	Mar	Abr	May	Jun
Traducción de libro “Use Case Driving Objeting Modelin – ICONIX.									
Estudio para la escogencia de la base de conocimiento.									
Apropiación del conocimiento acerca del uso de la herramienta gestora de contenido que genera la base de conocimiento.									
Modelamiento de un proyecto de la empresa para la captura de patrones de ejecución en el proyecto.									
Recolección de información acerca de la metodología de desarrollo manejada al interior de la empresa.									
Captura de información de la internet de procesos de desarrollo que permitiera alimentar los conocimientos del autor de este trabajo en estos temas.									
Montar la información definida para el proceso de desarrollo de la empresa en la base de conocimiento.									
Depurar la base de conocimiento.									
Socialización y retroalimentación de la base de conocimiento ante el personal de la empresa.									
Construcción de la Monografía.									
Construcción y preparación de la sustentación.									

ANEXO C. RECURSOS DISPONIBLES

Andrés Sánchez Comas, estudiante de último semestre de Ingeniería Electrónica de la Corporación Universitaria de la Costa el cual inicio el proyecto de grado mientras laboraba como estudiante en práctica de la empresa Bermit Ltda.

Rubén Sánchez Dams, Ingeniero Electrónico; director y asesor del proyecto de grado. Estudiante de Maestría de Ingeniería de Software en la Universidad del Norte. Líder de Grupo de Investigación y Docente Tiempo Completo del programa de Ingeniería Electrónica de la Corporación Universitaria de la Costa CUC.

BERMIT Ltda., empresa especialista en sistemas inteligentes de transportes, automatización, desarrollo de software y desarrollos tecnológicos a medida ubicada en Barranquilla-Colombia, fue la empresa guía y patrocinadora de este proyecto de grado que además de suministrar la información necesaria para la construcción de la base de conocimiento en EPF Composer permitió el uso de dos integrantes de la empresa que manejan ampliamente el tema de las metodologías de desarrollo dentro de la organización:

Marco Ebratt Orozco, Ingeniero de Sistemas. Ingeniero de Proyectos con 11 años de experiencia en el área de proyectos y que ha desarrollado proyectos en el área de Tecnología de la Información

Heyder Páez Logreira, Ingeniero Electrónico. Ingeniero de Desarrollo con año y medio de experiencia en el manejo de proyectos y buen

conocimiento en metodologías de desarrollo habiendo participado en tres proyectos de desarrollo tecnológico dentro de la empresa hasta la fecha.

Harry Morales, Ingeniero Electrónico. Ingeniero Desarrollo con año y medio de experiencia en el desarrollo de dispositivos electrónicos habiendo participado en tres proyectos de desarrollo tecnológico dentro de la empresa hasta la fecha.

Además del recurso humano el presente proyecto de grado se basó de sistemas computacionales para el registro, organización, análisis y procesamiento representado en un computador portátil perteneciente al autor de este trabajo. Dicha información constituye un recurso importantísimo y que en su mayoría fue suministrada por los Ingenieros de la empresa en cuanto a bibliografía de procesos de desarrollo de proyectos, informes, propuesta de proyectos, información de proyectos, y anotaciones de los proyectos realizados hasta la fecha por el personal respectivo de la empresa. Y por último toda la información disponible que hay en los sistemas de información del internet que sin duda aportaron un valor significativo al desarrollo del proyecto.